commodore

COMPUTER CLUB

17

L. 3.000

La rivista degli utenti di sistemi Commodore

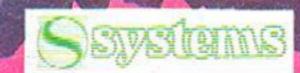
Mensile - febbraio/marzo 1985 - Anno IV - N. 18 - Sped. Abb. Post. Gr. III/70 - CR Distr. MePe

Traiettoria satelliti

Hard copy HI-RES 64

Collisioni Sprite fondo

Kernal II

Raster Register

F. Amatori

20 programmi didattici che offrono, assieme al divertimento, una chiara, esauriente ed appassionante guida alla programmazione in BASIC.
Non più la cieca e passiva digitazione di programmi trovati qua e là sulle riviste, senza riuscire a capire la logica ed il linguaggio della macchina: per la prima volta, attraverso l'esame dettagliato della costruzione del programma (analisi del problema, diagrammi di flusso, traduzioni in BASIC, listati), chi possiede il VIC 20 può acquisire finalmente, senza fatica e nel modo giusto, la capacità di usarlo per elaborare con entusiasmante disinvoltura altri programmi utili e divertenti, per mettere alla prova la propria creatività e fantasia.
Perché una macchina è utile solo se ci "serve" e se impariamo ad usarla nel modo giusto.

I 20 programmi descritti ed illustrati nel volume sono disponibili su una cassetta che potrà essere richiesta versando l'importo di L.12.600 sul c.c.p. n. 135004 intestato a Armando Curcio Editore - Servizio Clienti, specificando la causale del versamento.

ARMANDO CURCIO EDITORE

Sommario

RUBRICHE

Direttore: Alessandro de Simone
Redazione/collaboratori: Giovanni Bellù, Andrea e Alberto Boriani, Giancarlo Castagna, Eugenio Coppari, Marco De Martino, Giancarlo Mariani, Enrico Scelsa, Fabio Sorgato, Danilo Toma
Segreteria di redazione: Maura Ceccaroli, Piera Perin
Impaginazione/Illustrazioni: Francesco Amatori, Renato Caruso
Direzione, redazione: V.le Famagosta, 75 - 20142 Milano - Tel. 02/8467348
Pubblicità: Milano: Mirco Croce (coordinatore), Michela Prandini, Giorgio Ruffoni, Claudio Tidone, Villa Claudio - Segreteria: Lilliana Degiorgi
● Roma: Spazio Nuovo - via P. Foscari 70 - 00139 Roma - Tel. 06/8209679
Abbonamenti: Marina Vantini
Tariffe: prezzo per copia L. 3.000. Abbonamento annuo (11 fascicoli) L. 28.000. Estero: il doppio. Abbonamento cumulativo alle riviste Computer e Commodore Computer Club L. 55.000.
I versamenti vanno indirizzati a: Systems Editoriale Srl mediante assegno bancario o utilizzando il c/c postale n. 31532203 intestato a C.C.C.
Composizioni: Compfoto - P.zza Bonomelli, 10 - Milano
Fotolito: Systems Editoriale Srl
Stampa: La Litografica S.r.l. - Busto Arsizio
Registrazione: Tribunale di Milano n. 370 del 2/1/82 - Direttore Responsabile: Michele Di Pisa
Sped. in abb. post. gr. III - Pubblicità inferiore al 70%
Distribuzione: MePe, via G. Carcano 32 - Milano

Da questo mese la rivista C.C.C. è disponibile anche su nastro

Programmi su cassetta

☐ **Invece dei soliti giochi, non potreste mettere sulla rivista su cassetta Commodore Club programmi di Utility e, comunque, diversi da ciò che si trova in giro? (*R. Ligabue - Roma*).**

● Come avrai già notato, da un po' di tempo la nostra rivista su cassetta provvede ad accontentare quella fascia di utenza che vuole qualcosa di diverso (e di meglio) da far girare sul proprio personal. A tal proposito, hai visto (o meglio, sentito) il programma LA VOCE (L. 10.000 supplemento speciale a Commodore Club N. 3), che fa parlare il Commodore 64?

POKE PEEK & SYS

☐ **Potreste pubblicare i Poke e le SYS e le PEEK più usate sul C-64, C-16 e Vic 20? (*L. Lorenzini - Campogalliano. P. Baldetti - Roma. C. Veronese - Alessandria*).**

● L'altra rivista che pubblichiamo (dal semplice titolo Commodore, L. 3.000), pubblica fin dal N. 4 (ottobre '84) le rubrica "Cornucopia" e "Microscopio", che contengono numerosissime microroutine, descrizioni delle varie locazioni di memoria, loro utilizzo ed altre cose del genere.

Anche noi, come puoi aver notato, dedichiamo ormai in modo sistematico parecchio spazio a tale interessante argomento.

Listati all'indietro

☐ **È possibile realizzare un programma in L.M. per poter listare i programmi Basic anche "all'indietro"? (*A. Pizzulli - Freinfeld*).**

● Con un computer si può fare tutto e, di conseguenza, anche la routine che proponi. Ma, santo cielo, ti rendi conto che tutte le volte che accendi il computer devi caricare tale utility e ricaricarla tutte le volte che resetti il tuo Commodore?

Io, personalmente, non ho mai avvertito l'esigenza di listare le linee basic in ordine inverso. Se ti interessa esaminare solo un gruppo di linee, è sufficiente digitare, ad esempio:

LIST 450-789.

Computer & HI-FI

☐ **Posso collegare il computer al mio impianto hi-fi con il cavo che normalmente collega al TV il Commodore 64?**

● Il collegamento con un impianto hi-fi è possibile a patto che si rispettino le connessioni riportate sul libretto di istruzioni del calcolatore.

Nell'appendice che si riferisce alle connessioni I/O (input-output) il collegamento si realizza collegando il cavetto di massa con GND e quello audio con AUDIO OUT. È ovvio che il segnale che esce è monofonico (e non stereo), e come tale deve esser trattato.

Collegamento con Tv moderni

☐ **È possibile collegare un C-64 direttamente attraverso la presa "euro" (o stat) del Tv in modo da avere un'immagine, se possibile, migliore? (*L. Piccinato - Solaro*).**

● I moderni televisori sono dotati ormai di un connettore che consente il collegamento cui accenni in modo da "bypassare" lo stadio di alta frequenza. Per fare ciò è necessario rintracciare, sul libretto di istruzioni del TV, i piedini (pin) contrassegnati con GND (massa) Video PAL Input (Ingresso Vide) e Audio Input (Ingresso Audio) e collegarli, rispettivamente, mediante cavetto bipolare schermato, alle prese GND, Video Out e Audio Out del computer.

Coloro che sono in grado di effettuare il collegamento, sanno già come operare e la tua domanda, al contrario, mi fa intuire che non sei molto esperto nell'effettuarli. Ti sconsiglio, quindi, di intervenire. Rivolgiti ad un bravo tecnico TV che, con modica spesa, dovrebbe essere in grado di rifornirti del cavetto di cui hai bisogno.

Strani segni nei listati

☐ **In alcuni listati di altre riviste trovo, tra le altre istruzioni, PRINT [4 DOWN] oppure PRINT [3 LEFT]. Che cosa significano? (*E. Guida - Gaeta*).**

● [4 DOWN] significa che è necessario frapporre, tra virgolette (") quattro caratteri speciali ottenibili premendo il tasto di CRSR DOWN (il secondo da destra in basso). Il modo originale di indicare i caratteri speciali dovrebbe servire per evitare dubbi di interpretazione nei programmi che li contengono. Purtroppo ne fanno sorgere altri, come quello che esponi. Altri messaggi normalmente usati sono: [CLR] Cancella schermo. [HOME] Posizionamento del cursore in alto a sinistra. [BLK] Colore nero. [F1] Tasto funzione F1, eccetera.

Programmi errati?

☐ **Alcuni programmi che ho digitato dalla vostra rivista non girano. Come mai? (*M. Alani - Fidenza*).**

● La tua domanda è un po' generica e non ci consente di esserti di aiuto in nessun modo. Tieni presente che tutti i programmi pubblicati vengono sempre provati prima di pubblicarli ed eventuali anomalie sono prontamente rese note.

È molto probabile, quindi, un tuo errore nel digitarli.

Computer per bambini

☐ **Tra i programmi di matematica non ve n'è uno dedicato ai bambini. Con la buona grafica del C-64 si potrebbe realizzare un programmino che illustri i concetti con semplici diagrammi di Venn. (*G. Cardone - Napoli*).**

● La tua richiesta è più che legittima. Purtroppo, i programmi di didattica elementare richiedono la competenza e la professionalità di psicologi e maestri elementari che sono gli unici che possono esser considerati esperti nell'affrontare tali argomenti. Realizzare un programma realmente utile per fini didattici è molto più complesso di quanto possa sembrare a prima vista. È infatti indispensabile tener conto della particolare psicologia dell'utente (bambino) e della possibilità di intervento da parte dell'educatore. Un programma di tal genere non è affatto un "programmino", ma il risultato di un lavoro enormemente lungo ed articolato. Che, in ogni caso, non può esaurirsi con interventi di carattere episodico.

Nonostante tali premesse molti si sono cimentati addirittura nel proporre nuovi linguaggi (Logo, Turtle graphics) che tengono conto delle esigenze da te giustamente esposte. Purtroppo tali linguaggi, e la relativa manualistica e letteratura, sono in lingua inglese e non molto diffusi nel nostro Paese.

Tra un po' di tempo vedrai una nostra iniziativa in tal senso e, anzi, ti invito fin da adesso a tenere gli occhi aperti per sperimentarla. Di che si tratta? Un po' di pazienza, diamine!

Informazioni sui programmi

☐ **Alcuni lettori chiedono informazioni riguardanti software posto in vendita da alcune ditte con particolare riferimento alla compatibilità, affidabilità, velocità, eccetera.**

● Non ci è possibile rispondere, perché il software in vendita ha ormai raggiunto una vastità incredibile e provare tutti i programmi, e farsene un'opinione corretta, è pressoché utopistico.

Pubblicheremo, di tanto in tanto, a patto di avere un'incoraggiamento da parte dei lettori, le prove di programmi di interesse generale, soprattutto sull'altra nostra rivista Commodore. Per ciò che riguarda le informazioni tecniche, rivolgetevi alle ditte che commercializzano il prodotto che interessa.

Corsi di lingua su computer

☐ **Mi interessa molto un corso di lingua inglese su Commodore 64.**
Vorrei sapere se esiste un programma del genere e dove trovarlo. (*G. Nicoletti - Pedace (CS)*).

● Non ci risulta che esista un programma del genere a meno che non sia il solito vocabolario bilingue. Rivolgiamo - quindi - la domanda a chi ne sa qualcosa, pregandolo di farci pervenire informazioni a riguardo.

Dubitiamo, però, che possa esistere un efficiente corso su computer perché per imparare una lingua sono più importanti una corretta pronuncia e, soprattutto, la conversazine. In che modo un computer possa risolvere questi problemi, ai livelli attuali della tecnologia, proprio non riusciamo ad immaginarlo.
Se vuoi imparare l'inglese, recati per un mesetto a Londra.
Ora come ora è l'unico sistema per imparare una nuova lingua.

Commodore 64 & C-16.

☐ **Posseggo un Commodore 64, ed ho intenzione di acquistare un C-16. C'è la possiblità di far lavorare i due computer contemporaneamente sullo stesso video e con le stesse periferiche? (*S. Damiano - Palermo*).**

● Le periferiche che puoi usare col C-16 sono le stesse del Commodore 64 e del Vic 20 (drive 1541, stampanti ecc.) ad eccezione del registratore e dei joystick (e paddle) che hanno un diverso connettore.

Non riesco, invece, a capire il vantaggio di far apparire sullo stesso video l'output di due computer. In linea di massima, comunque, quanto chiedi è possibile a patto di realizzare un miscelatore UHF in grado di "scaricare" (a meno di eventuali in-terferenze) i due segnali sullo stesso TV. La frequenza di uscita del C-16 è identica a quella del Commodore 64.

Totocalcio

☐ **Sarei interessato ad acquistare il programma del totocalcio per il Commodore 64, ma vorrei un parere. (*A. Cristofanelli - Fabrica di Roma (VT). G Meneguzzo - Roma*).**

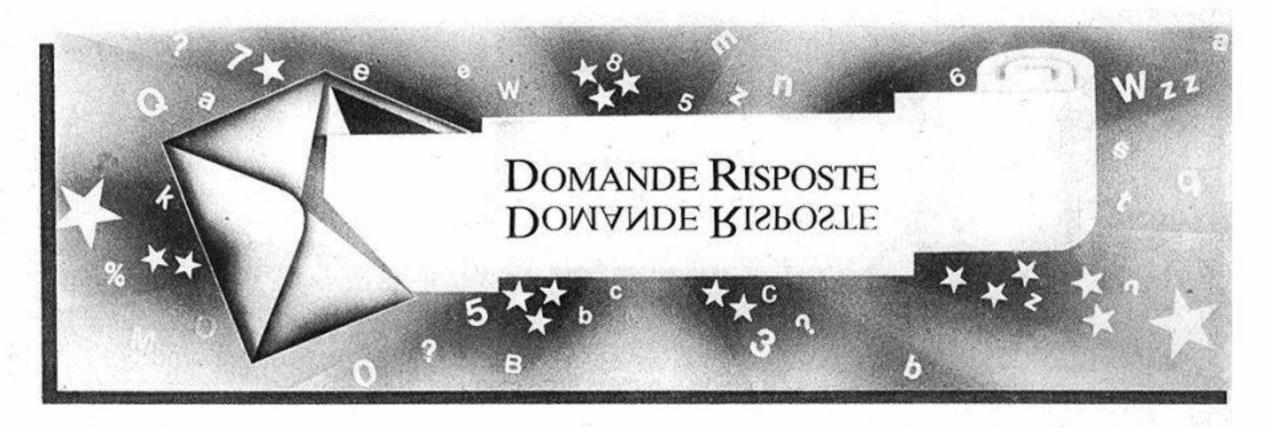
● Non specifichi di quale programma vuoi il parere e ci è impossibile risponderti a causa dell'enorme varietà di programmi Toto posta in vendita. Tieni presente, comunque, che previsioni del gioco del calcio sono praticamente impossibili, anche servendosi di un computer. Se esistesse un programma in grado di far 13 non verrebbe di certo posto in vendita dagli autori, che impiegherebbero diversamente il frutto del proprio lavoro.

I programmi cui ti riferisci non fanno altro che attribuire alcuni parametri alle squadre di calcio, al campo di gioco, al pubblico dei tifosi, alle condizioni atmosferiche, alla presenza o assenza di alcuni giocatori della squadra eccetera. Il risultato dell'elaborazione consiste nell'individuare la probabilità di vittoria più elevata per le diverse squadre in campo. Le procedure per tale determinazione sono a volte molto fantasiose e, spesso, decisamente poco scientifiche, se non addirittura errate.

Colori e caratteri del 64

☐ **Mi hanno detto che il Commodore 64 può riprodurre un numero di colori maggiore dei 16 comunemente ottenibili da tastiera come pure altri caratteri semigrafici. In che modo? (*C. Welker - Modena*).**

● I colori riproducibili sono soltanto 16. Particolari tecniche di programmazione in Linguaggio Macchina, ed in particolare ricorrendo all'interrupt, consentono di ottenere, in certi casi, altri colori. Se, ad esempio, un carattere su fondo blu viene colorato, alternativamente, in bianco e in nero, è evidente che otterremo la visualizzazione dello stesso carattere. Colorato però in grigio.

Ciò è però possibile lavorando ad alta velocità e approfittando della permanenza dell'immagine sulla retina dell'occhio umano. Sei comunque sicuro che i 16 colori standard non sono sufficienti per le tue esigenze?

Per ciò che riguarda la possibilità di visualizzare altri caratteri, ciò è possibile ricorrendo alla programmazione degli stessi caratteri. Ti consigliamo di leggere l'articolo "Ridefinizione dei caratteri" pubblicato sul N. 16 di C.C.C.

Riparazioni

☐ Il mio Commodore 64 si è già rotto un paio di volte e lo hanno trattenuto parecchio tempo prima di riconsegnarlo riparato.
Come mai il tempo necessario alle riparazioni è così lungo? (*A. Orlandi - Roma*).

● Alcuni circuiti integrati dei computer Commodore sono fabbricati esclusivamente dalla Commodore e, a quanto pare, è necessaria una lunga pratica burocratica per ottenerli.

Ciò, comunque, non giustifica completamente i ritardi (a volte di mesi interi) nella riconsegna del materiale inviato in riparazione. Siamo infatti completamente solidali con i lettori cui capitano tali disgrazie e, anzi, rivolgiamo ai responsabili della Commodore le lamentele degli sfortunati utilizzatori.
Saremo ben felici di pubblicare la statistica riguardante la percentuale (sul venduto) di apparecchi inviati in riparazione, la giacenza ed il costo medio delle riparazioni effettuate al di fuori della garanzia.

Color sul Vic 20

☐ Sul mio Vic 20 non riesco a riprodurre i colori: arancio, marrone, rosa, grigio1, grigio2, grigio3, verdino e celeste. Come mai? (*L. Barbetta - Milano*).

● Tali colori sono riproducibili solo sul Commodore 64 (sul C-16 sono comunque riproducibili 16 colori). Per esempio, con:

PRINT CHR$ (144) "COMPUTER"

la parola "computer" compare in nero sia sul Vic 20 che sul Commodore 64. Se invece digiti:

PRINT CHR$ (155) "COMPUTER"

col Vic 20 la parola Computer verrà visualizzata nello stesso colore che aveva precedentemente il cursore, mentre col Commodore 64 verrà riprodotta in grigio3. Il Vic 20, in altre parole, si limita ad ignorare il carattere di colore che per il Commodore 64 ha invece un ben preciso significato.

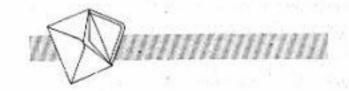
Caratteri speciali

☐ Vorrei notizie su come ottenere i caratteri speciali del Commodore 64. In particolare che simbolo è quello che compare spesso nei vostri listati e di cui allego un esempio? (*A. Rifulco -Roma*).

● Il carattere misterioso (una "Q" in reverse) è un CRSR DOWN che si ottiene col tasto Cursor Down (Cursore in basso). Ti consiglio, ad ogni buon conto, di leggere l'articolo "I caratteri speciali del computer Commodore" pubblicato sul N. 8 di C.C.C.

Microprocessore 6510

☐ Quali funzioni può eseguire il micro 6510? (*I. Pintori - Roma*).

● Il 6510 è il microprocessore montato sul Commodore 64 ed è praticamente identico al 6502 montato normalmente sul Vic 20 e sui PET. L'Assembler per i due micro è pertanto lo stesso e gli articoli sul linguaggio macchina che vedi pubblicati valgono per tutta la "famiglia" 65XX.

Applicazioni per la user port

☐ È possibile utilizzare la User Port per comandare apparecchi esterni? (*F. De Colle - Civitavecchia*).

● Sulla rivista Commodore (N. 3) è stato addirittura pubblicato il circuito stampato di un simile accessorio. Puoi richiedere il numero arretrato (L. 5.000) specificando che si tratta di Commodore (e non di C.C.C.) alla nostra Redazione - Servizio arretrati.

Una POKE misteriosa

☐ Su molti listati compare spesso un POKE 198, N (con N diversa di volta in volta). A che serve? (*A. Mocellin - Termine di Cassola*).

● È una locazione di memoria che riguarda il Buffer di tastiera. Con tale poke, e con altre dieci, è possibile realizzare tecniche di programmazione interessanti che sono state ampiamente descritte sul N. 8 di C.C.C. nell'articolo "Il buffer di tastiera".

Registratori Vic 20 & C-16

☐ Sarà possibile collegare il registratore del Vic 20 (e del Commodore 64) al C-16? (*G. De Vanna - Palese (BA)*).

● I due registratori sono perfettamente eguali e si differenziano solo nel connettore che è di tipo nuovo, non unificato e, per quanto ne sappiamo, prodotto solo dalla Commodore. Non sappiamo se connettori di tal tipo saranno posti in vendita "sfusi" ma, ne siamo sicuri, prima o poi qualcuno penserà a risolvere il problema con un classico uovo di Colombo, annullando il tentativo di "monopolio" che (siamo cattivi) si è tentato di introdurre col nuovo computer.

Compatibilità con i "grossi" Commodore.

☐ Il software del Commodore 64 è compatibile con i computer serie 8000 e 8096 SK? (*M. Giustino - Napoli*).

● Se si tratta di programmi in Basic, la compatibilità è completa tranne che in tre casi: POKE, PEEK, SYS. Se, in altre parole, nel programma sono presenti tali istruzioni, è indispensabile esaminare i loro argomenti (=

memorizzazione, su supporto magnetico, della variabile "A" seguita dal carattere CHR$ (13).

Un errore che alcuni commettono consiste nel digitare il carattere "13" dopo un dato SENZA il punto e virgola. Col seguente comando, infatti,...

PRINT#1, A: PRINT#1, CHR$(13)

... sul supporto magnetico verranno trascritti, oltre al dato numerico "A", TRE caratteri di ritorno carrello CHR$ (13).

Non penso, però, che gli errori cui ti riferisci siano dovuti a tali imprecisioni ma ad una dimenticanza, comune a chi lavora con i file, di cui ora parliamo.

Il comando INPUT (sia sotto la forma INPUT che sotto la forma INPUT#) accetta un massimo di 80 caratteri. Col comando PRINT (e con l'analogo PRINT#) è invece possibile dare il comando di stampare stringhe di una lunghezza massima di ben 255 caratteri. Se si ricorre al carattere di punto e virgola è addirittura possibile generare una stringa di caratteri tanto lunga da riempire l'intero dischetto (o nastro). Esempi:

```
...
100 FOR I = 0 TO 255
110 A$= A$ + "X": NEXT
120 PRINT#1, A$
...
450 PRINT#1, "X"; : GOTO 450
...
```

Se sul supporto magnetico viene registrata una stringa come A$, ottenuta con le righe 100-110, oppure una stringa molto più lunga (riga 450), ad un successivo comando INPUT# queste non potranno esser rilevate ed il drive si bloccherà con il LED lampeggiante. L'errore che ne risulta è STRING TOO LONG ERROR. L'unico modo di leggere tali stringhe è quello di ricorrere al GET# che legge un carattere per volta.

Match C-16 e Cbm-64

☐ **Il C-16 è migliore del mio Commodore 64? (*R. Sant - Treppo Grande*).**

● Il C-16 è un computer nato per sostituire il Vic 20 che è rimasto sul mercato per un tempo straordinariamente lungo considerando che si tratta di un personal computer. Il Vic 20, nonostante l'incredibile successo conseguito, manifestava infatti gli acciacchi della vecchiaia (ben quattro anni di onorata produzione) e non rappresentava più quelle caratteristiche che gli avevano consentito di snobbare la concorrenza giapponese ed (in minima parte) europea.

La maggior quantità di RAM di cui è dotato il C-16, la presenza (finalmente) dei comandi basic per il suono e la grafica in alta risoluzione, pongono il nuovo nato della Commodore, considerando il prezzo al pubblico, un osso spaventosamente duro per i giapponesi che si sono affacciati da poco con la proposta MSX. Con quest'ultimo standard, infatti, è necessario spendere più del doppio per venire in possesso di un sistema che svolga le stesse funzioni. Il C-16, però, non disponendo dell'uso degli sprite e del favoloso circuito integrato generatore di effetti sonori (S.I.D.), ben difficilmente potrà essere utilizzato da quegli utenti che, a basso prezzo, desiderano usufruire di potenzialità che sono ancora da considerare eccezionali nel Commodore 64.

Il futuro del C-64

☐ **Sto per acquistare il Commodore 64. Pensate che resterà "il Re" dei personal anche per il futuro? (*F. Bonanni - Terni*).**

● Nessun computer, nemmeno il favoloso Commodore 64, potrà mai essere immune dall'obsolescenza. Anche se ha tutte le carte in regola per restare in auge per un periodo ancora molto lungo (migliaia di programmi disponibili, sprite, effetti sonori, basso prezzo ecc.) prima o poi è destinato ad esser sostituito da altre "diavolerie" che saranno capaci di inventare. Lo stesso destino è toccato, tocca, e toccherà a qualsiasi computer di qualsiasi marca.

Se non acquisti il computer per paura di ritrovarti, fra qualche anno, un apparecchio "vecchio", commetti però un grosso errore: rischi di rinviare la conoscenza della programmazione che si può ottenere soltanto adoperando a casa propria un computer.

Ciò rappresenta, coi tempi che corrono, un suicidio culturale, dato che un qualsiasi posto di lavoro richiederà, anche nell'immediato futuro, le conoscenze basilari dell'informatica.

L.M. e linguaggio Assembler

☐ **Che differenza c'è tra linguaggio macchina e linguaggio Assembler? (*L. Fiorentino - Napoli*).**

● Il linguaggio macchina è un linguaggio di basso livello utilizzante, normalmente, la notazione esadecimale che, invece di utilizzare dieci simboli numerici (da "0" a "9"), ne utilizza 16: da "0" a "9" e da "A" ad "F". Volendo, ad esempio, memorizzare il dato "08" nella locazione RAM il cui indirizzo è (in esadecimale) A000, la procedura da seguire consiste nel caricare il dato "08" in un registro di memoria (chiamato accumulatore) e nel trasferire il suo contenuto (appunto, "08") nella locazione di indirizzo A000.

Lavorando in linguaggio macchina la sintassi corretta è la seguente:

```
A9 08 00 A0 60
```

Come puoi notare, a meno che non "mastichi" di già il linguaggio macchina, tale gruppo di dati ed istruzioni sono ben poco comprensibili. Per render più chiaro il programma si ricorre ad un linguaggio più evoluto, l'Assembler, che utilizza, come istruzioni, gruppi di caratteri che "ricordano" l'effetto dell'istruzione stessa. Lo stesso gruppo di istruzioni di prima sarà, in Assembler, così digitato:

```
LDA# 08
STA A000
RTS
```

LDA deriva da Load Accumulator; STA significa STore Accumulator; RTS significa Return from Subroutine. Questo linguaggio è un po' più comprensibile ma non certo chiaro soprattutto per la gran massa di utenti di personal computer.

A tale scopo viene adoperato il Basic che è un linguaggio ad alto livello e facilmente gestibile da persone dotate anche di modesta preparazione in informatica.

di cancellazione schermo viene attivata, esse non riescono ad eseguire l'ordine perché non sono riuscite a riceverlo.
I caratteri casuali che noti sullo schermo sono, pertanto, il contenuto di tali locazioni RAM che, "svegliatesi" quando la routine di cancellazione è stata ormai eseguita, contengono valori casuali. In conclusione: se il difetto sparisce premendo i tasti SHIFT e CLR (cancellazione schermo) e non si verifica fino a che non si spegne l'apparecchio, l'inconveniente non è grave ed è dovuto ad un ritardo del raggiungimento del regime di funzionamento delle RAM. Se, al contrario, capita durante il funzionamento, è bene far effettuare una verifica.

Le SYS di partenza

☐ **Di alcuni programmi in Linguaggio Macchina registrati su nastro ho dimenticato l'indirizzo di partenza che consente di attivarli mediante SYS. C'è un sistema per individuarlo? (*S. Fois - Forlì. F. Campogrande - Milano*)**

● Purtroppo no. A differenza dei programmi basic in cui, normalmente, la prima linea ad esser eseguita è sempre quella a numerazione più bassa e cioè quella "in testa" al programma, in una routine in L.M. non sempre la prima locazione rappresenta la prima istruzione da eseguire. Molti autori di routine seguono altri criteri.
Sebbene, ad esempio, una routine occupi le locazioni da 49152 a 50130, è probabile che la SYS di partenza sia stata fissata a 50000 per fare in modo da ricordarla più facilmente. Alle locazioni numerate da 49152 corrisponderanno, magari, alcuni byte che rappresentano caratteri ASCII che formano un messaggio. Digitare SYS 49152 non porta, in tali casi, alla corretta inizializzazione della routine ma, molto spesso, all'inchiodamento del computer.
Fortunatamente non tutti seguono questa moda e, spesso, il primo byte della routine in Linguaggio Macchina rappresenta anche la prima istruzione da eseguire. Per conoscere l'esatto indirizzo di partenza di routine L.M. che vanno caricate con la sintassi ",1" (per il registratore a cassette) è sufficiente posizionare il nastro in corrispondenza dell'inizio della routine e digitare, semplicemente, OPEN1
Dopo un pò di tempo il computer fermerà il motore del registratore e apparirà il consueto READY (anche se si tratta di un nastro "protetto"). A questo punto, digita:
PRINT PEEK (829) + PEEK (830) *256
Il valore che verrà visualizzato rappresenta l'indirizzo del primo byte. Mentre:
PRINT PEEK (831) + PEEK (832) *256
ne rappresenta l'ultimo. Tale procedura può esser utile anche per individuare l'inizio e la fine di programmi Basic.

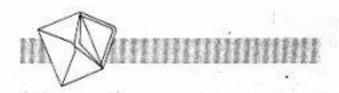
Parentesi quadre

☐ **Quando uso espressioni matematiche contenenti parentesi quadre, il computer risponde con SYNTAX ERROR. È un difetto del computer? (*L. Colli - Milano*)**

● Assolutamente no. Le parentesi tonde, quadre e graffe cui siamo abituati a scuola sono state introdotte dall'uomo al fine di ottenere una migliore schematizzazione grafica. Il computer, qualunque esso sia, non ha tali esigenze e riconosce solo le parentesi tonde. Per realizzare una "nidificazione" di parentesi ricorri pure a più livelli di parentesi tonde. Per calcolare la prima radice di un'equazione di secondo grado puoi, ad esempio, utilizzare la seguente formula:
X1 = (-B+SQR (B*B-4*A*C))/(2*A)

Meglio del 1541

☐ **Esiste un drive compatibile con i computer Commodore che costi meno di quello originale 1541? (*R. Francavilla - Foggia*)**

● Il 1541 è un drive intelligente. Nel senso che, oltre a possedere una memoria RAM, dispone di un microprocessore e di un vero e proprio Sistema Operativo su ROM. Ciò rende tale periferica indipendente dal computer cui è collegata. Tanto è vero, che non è necessaria alcuna interfaccia per collegare il 1541 al Commodore 64, al Vic 20 oppure al nuovo C-16. Dubitiamo, pertanto, che un costruttore riesca a porre sul mercato un apparecchio di elevate caratteristiche elettroniche ad un prezzo più basso di quello Commodore.
Questo, non dimenticarlo, è determinato in gran parte dall'enorme produzione, che consente di abbassare il costo di fabbricazione ai livelli che ingiustamente ritieni elevati.

Presunte furbizie

☐ **Perché, invece di far riferimento ad articoli precedenti, non pubblicate nuovamente i listati che affermate indispensabili per digitare i programmi che pubblicate? (vedi le routine grafiche di Toma). È forse un modo per vendere gli arretrati? (*G. Perrella - Casalbore. S. Nepoti - Argenta*)**

● Se volessimo guadagnare di più ci potremmo limitare ad aumentare il prezzo di copertina approfittando dell'aumento del numero di pagine necessario per seguire il tuo consiglio. Come puoi notare molti sono ormai gli articoli che suggeriscono di rileggere scritti precedenti allo scopo di meglio assimilare gli argomenti trattati. Il lettore "medio" comprende ormai la necessità di considerare la nostra pubblicazione come una vera e propria enciclopedia a dispense. Lavori di notevole pregio, inoltre, resterebbero lettera morta se non se ne utilizzassero tali interessanti caratteristiche.

Da Sinclair a Commodore.

☐ **Esistono programmi che consentono di adattare i programmi dello Spectrum al Commodore 64? (*A. Perrone - Brindisi*)**

● Le due macchine sono completamente diverse. Non solo perché lo sono i due microprocessori, ma anche perché la sintassi, specialmente per ciò che riguarda la grafica ed il sonoro, è completamente differente.
In teoria una trasformazione potrebbe esser possibile attraverso una routine di "emulazione". In pratica, il tempo per realizzarla è tale che risulta decisamente più economico acquistare i due computer ed usare, a seconda dei casi, l'uno o l'altro.

segue a pag. 79

EDITORIALE · EDITORIA

ORIALE · EDITORIALE

Riprendiamoci il Computer

I calcolatori furono realizzati per consentire all'uomo di ottenere in breve tempo elaborazioni che, affidate altrimenti ad operatori "umani", avrebbero richiesto periodi di tempo inamissibili.

Con l'abbassamento dei prezzi, e la conseguente diffusione del personal computer, l'asse delle elaborazioni si è lentamente spostato dalla programmazione "seria" ai videogiochi.
Il guaio è che molti si sentono in diritto di associare al calcolatore domestico l'idea del passatempo, dell'alienazione e, troppo spesso, della nocività del nuovo apparecchio introdotto in casa.

Del resto la martellante pubblicità, quasi unidirezionale verso i gochi, e la notevole mole di cassette (ancora di giochi) presenti in edicola, autorizzano, purtroppo, a confinare il computer nell'area del gadget superfluo.

Fortunatamente c'è qualcuno che individua nel personal computer le potenzialità idonee per stimolare la fantasia, la ricerca scientifica o la semplice curiosità ed il desiderio di saperne di più.

Diamo, allora, al computer il ruolo che gli spetta e, lasciando da parte la voglia "insopprimibile" di annientare alieni e nemici, cerchiamo di considerare con più attenzione il "laboratorio di logica" che colleghiamo al TV domestico.

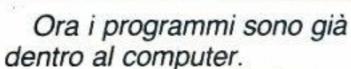
E se ti avvincono solo i giochi di avventura o di grande abilità, hai mai pensato che l'esplorazione della memoria e la modifica di puntatori è meglio del Camel Trophy? Oppure pensi di non esserne capace...

Alessandro de Simone

Entrare nella memoria del computer può essere un'avventura...

Lo straordinario C

L'unico con 4 p

Ora i programmi sono già dentro al computer.

Questo è il nuovissimo Plus/4. Il primo personal che ti dà, oltre a complete caratteristiche professionali, anche 4 programmi di altissima qualità già incorporati: un programma di WORD PROCESSING, per scrivere lettere e relazioni; un FOGLIO ELETTRONICO per la pianificazione finanziaria; un DATABASE, per la creazione e gestione di archivi; un pacchetto di BUSINESS GRAPHICS, per visualizzare i tuoi dati sotto forma di diagrammi e istogrammi.

È facile: premi un tasto e puoi richiamare subito i programmi che vuoi, senza dover aspettare il tempo di

WORD PROCESSING

FOGLIO ELETTRONICO

DATA BASE

commodore

ORDINAMENTO DI VETTORI

Una introduzione al problema, classico per un calcolatore, dell'ordinamento di valori numerici e alfanumerici

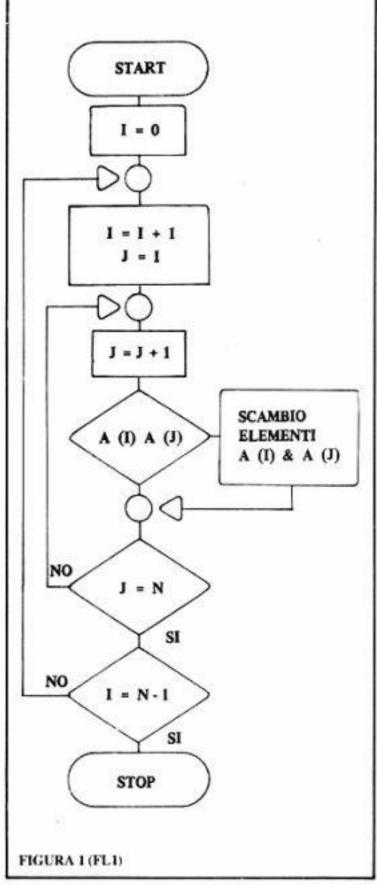
Un problema che spesso si pone al programmatore esperto come a quello alle prime armi è certamente quello del SORT. Con questo termine si indica il processo di ordinamento di una serie di numeri o di caratteri alfabetici, alfanumerici o grafici.

Immaginiamo di avere un mazzo di carte che dobbiamo ordinare con l'asso a sinistra poi il due, il tre, il quattro e così via fino al re. Per fare ciò, prima di tutto dovremo cercare il valore più basso e metterlo all'estrema sinistra della serie. Cercheremo quindi la carta di valore minore tra le rimanenti e la inseriremo nella seconda posizione, e rifaremo la stessa procedura per le altre carte.

La stessa logica usata per ordinare il mazzo di carte, può essere applicata al computer per effettuare ordinamenti di numeri, lettere o caratteri grafici, opportunamente inseriti in un vettore. Una caratteristica discriminante i vari tipi di SORT, è il tempo, che può diventare ragguardevole nell'ordinare una quantità notevole di dati.

Le due routine qui proposte effettuano l'ordinamento (SORT) di vettori. In base al carattere che si farà seguire al nome (nome +"%" oppure "$" o nessun carattere), si potranno effettuare ordinamenti su numeri interi, reali, o qualsiasi. Nelle figure 1 e 2 è possibile osservare i Flow-Chart delle due routine pubblicate. Come si può notare il secondo Flow-Chart non è altro che il primo, ampliato, grazie all'utilizzo di un ciclo terziario (T) di cui parleremo in seguito.

La logica utilizzata in questa procedura è la stessa delle carte. Si cerca il record contenente il valore minore e lo si scambia con il contenuto del record considerato. Passando ora all'analisi della prima delle due strutture, possiamo notare che nella variabile N è contenuto il numero dei re-

FIGURA 1 (FL1)

cord che compongono il vettore. Possiamo inoltre notare l'esistenza di due cicli, uno interno all'altro.

Dopo l'azzeramento della variabile I, troviamo il suo incremento (I=I+1). L'utilizzo di questo contatore si è reso necessario per indicare la posizione del record, oggetto di studio o meglio, di confronto. Questo, all'inizio del ciclo sarà uguale a uno, sarà cioè considerato come primo record.

Un secondo contatore, "J", partirà dalla posizione I+1 (nel caso rappresenta il secondo record), per giungere poi all'ennesima posizione. Confronteremo, ogni volta, il record di indice I con quello di indice J. Se, dal confronto, risulterà maggiore il primo (I), allora avverrà lo scambio dei contenuti dei records come segue: Nella variabile di comodo "H" verrà inserito il contenuto del record di indice I; in questo

record, a sua volta, verrà inserito il valore del record di indice J e in quest'ultimo, il valore di H.

Se, invece, dal confronto risultasse minore o uguale al secondo record (di indice J), il programma salterà al confronto della variabile J, alla quale si perviene comunque. Si confronta J con N (dimensione vettore), per controllare se sono stati considerati, come oggetto di confronto, tutti i records, numerati da I+1 a N, con quello di indice I.

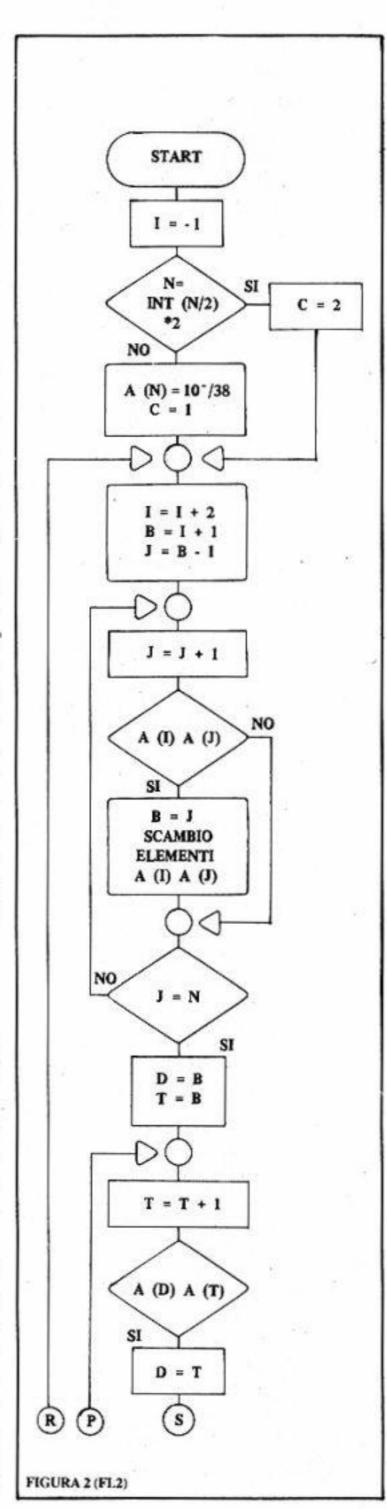
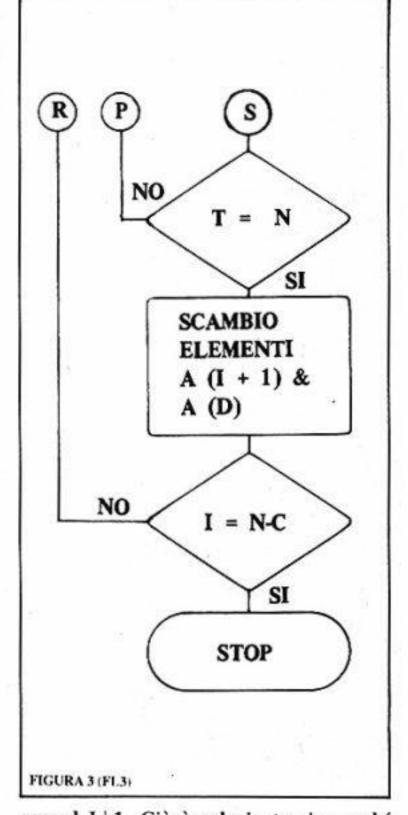
In caso positivo, siamo sicuri che nel record di indice I sarà contenuto il valore minore finora trovato e quindi questo record risulterà ordinato. In caso negativo, si tornerà ad incrementare J, per potere continuare i confronti. Un altro controllo segue questo appena analizzato: si tratta di quello tra I e N-1. Dopo avere ordinato il primo record, bisognerà passare ad ordinare il secondo incrementando I. Si passerà a considerare poi il terzo record, il quarto e così via fino al penultimo. Essendo la logica usata quella di confronto tra il record soggetto e i rimanenti, quando studieremo il penultimo record, ci accorgeremo che verrà confrontato solo con l'ultimo. In conseguenza di ciò, dopo l'eventuale scambio, risulteranno entrambi automaticamente ordinati. Come possiamo notare dalla figura 1, grazie alla scelta dello schema logico proposta, il numero dei confronti risulta decrescente.

Il primo listato

Il primo listato pubblicato rappresenta la stesura BASIC di quanto spiegato. I DATA considerati (47), sono presi casualmente digitati. In questo caso il tempo di esecuzione risulta essere di circa 16 secondi, piuttosto elevato, se consideriamo che possono verificarsi casi in cui è necessario ordinare centinaia di elementi.

Come possiamo rilevare, ogni volta che si effettua uno scambio, nel record I si trova il valore più piccolo trovato fino a quel momento, mentre nel record J viene trascritto il secondo valore. Di conseguenza, quando sarà finito un ciclo di J, nell'ultimo record oggetto di scambio (di indice J), sarà presente il secondo valore minore, quindi potremmo inserire questo valore nel

FIGURA 2 (FL2)

FIGURA 3 (FL3)

record I+1. Ciò è solo in teoria perché andrebbe bene nel caso in cui il record oggetto ultimo di scambio fosse proprio l'ultimo. Vediamo perché (esempio 2). L'ultima volta che si è effettuato uno scambio, è stato in 2; il valore scambiato con 0, è stato 5, il quale si è spostato nel quarto record. In teoria potremmo mettere al valore 5 nel record I+1, ma nel vettore è presente un valore minore di questo e precisamente 2 (nel record 6) il quale, evidentemente, non è stato giudicato minore di 0.

A questo punto potrebbero cominciare i confronti per l'ordinamento di I+1, non da I+1, bensì da 4 (indice del minor valore trovato prima di I). Se noi, la variabile B, indichiamo, ogni volta che si è effettuato uno scambio, la posizione di J nel quale è avvenuto, alla fine di questo ciclo (J), troveremo, in questa variabile, il punto di partenza di un terzo ciclo (T) di confronti, che verrà incrementato fino a N.

In un'altra variabile che chiameremo D,

converrà inserire il valore di B prima dell'inizio del ciclo terziario, in quanto ora si userà la tecnica, non più dello scambio di valori, ma di inserimento in questa variabile (D) del valore T, ogni qual volta si presentasse nel record T un valore minore rispetto al record D. Alla fine del ciclo citato prima (T), nel record D sarà presente il vero valore minore il quale andrà inserito in I+1. Si passerà quindi ad incrementare I. Come si può notare, l'incremento di I è 2; questo perché usciti dai cicli J e T, risulteranno ordinati i record I e I+1. Da ciò nasce la necessità di avere un vettore avente un numero pari di record. Il problema, comunque, non sussiste perché, come possiamo notare dal programma BASIC N.2, nel caso di un vettore dispari, un altro record verrebbe aggiunto al vettore, evidentemente riempito con valore talmente alto da non essere preso in considerazione. Questo discorso è valido anche per vettori stringa per i quali sarà necessario, oltre aggiungere al nome il carattere "$", anche riempire l'eventuale record aggiuntivo con una serie di CHR$ (255).

In questa spiegazione, è stato preso in considerazione il solo ordinamento crescente ma, con opportune modifiche, si può realizzare quello decrescente. Risulterà necessario cambiare i segni dei confronti e riempire l'eventuale record con un valore bassissimo (un CHR$ (0) se stringa, 10↑-38 se reale e -32767 se intero).

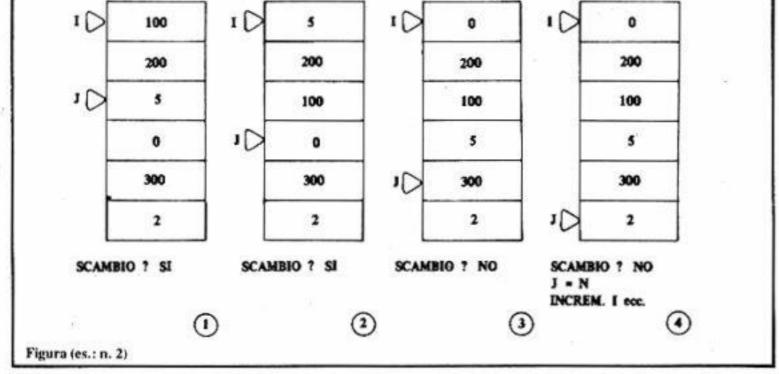
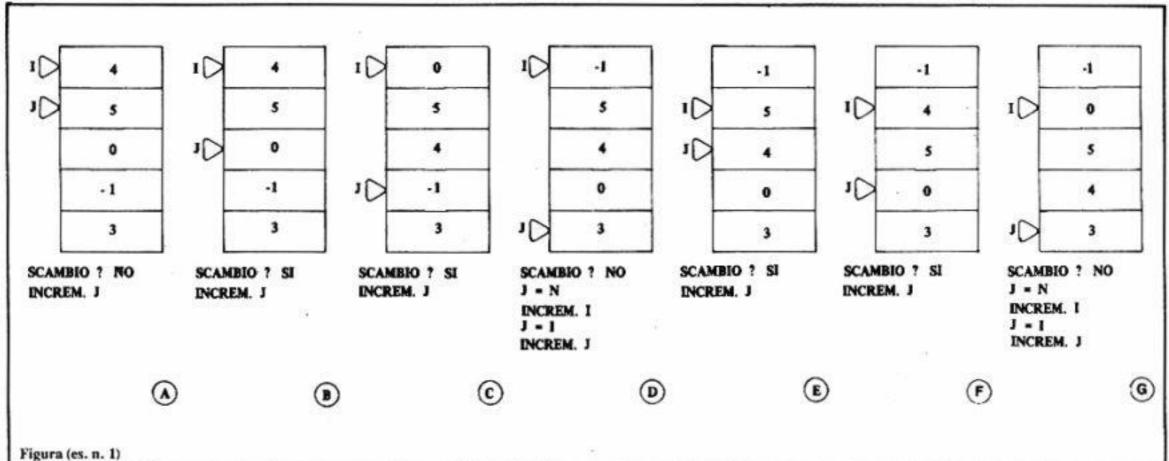
Passiamo ora all'analisi delle stesure BASIC di quanto sopra descritto: prederemo il considerazione per prima la routine del programma N.1. Alla riga 70 troviamo, dopo un CLR iniziale, la richiesta nel numero di records contenuti nel vettore, opportunamente seguita dal dimensionamento di questo.
Alla riga 80, troviamo un primo ciclo I, necessario al riempimento dei vari records; è opportuno o limitare il dimensionamento a questo valore (-1 perché in BASIC si parte dal record 0 non 1), o aumentare il numero dei DATA affinché siano sufficienti. Alla fine del ciclo è presente un azzeramento della variabile del tempo TI$. Ciò si è reso necessario per valutare il tempo di ordinamento impiegato dalla routine.

Nel caso in cui si usi procedura come subroutine, le righe fino alla 80, andranno cancellate in quanto il vettore, evidentemente, sarà già stato dimensionato e riempito. Con la riga 90 inizia ad operare la routine vera e propria. Comincia con un ciclo primario I, che assumerà valore 0 e verrà incrementato fino a N-2, perché, come specificato prima, lavorando in BASIC, bisogna decrementare gli indici di 1 (0<=1 e N-2<=N-1 spostamento indici).

Un secondo ciclo segue quello di I: Si tratta di J, che varierà da I=1 a N-1. Questo rappresenta il ciclo secondario.

Alla riga 100, troviamo il confronto tra i records di indice I e J. Nel caso di inferiorità di valore del primo, si salterà alla riga 120 (NEXT); in caso contrario, si passerà alla successiva riga 110 nella quale verrà effettuato lo scambio di valori mediante l'utilizzo della variabile intermedia H. Un NEXT introduce la riga 120; si tratta della fine dei cicli J e I, in ordine. Di seguito viene visualizzato il tempo necessario per effettuare l'ordinamento e, infine, il ciclo di stampa degli elementi ordinati che si conclude alla successiva riga 130 con la

Figura (es.: n. 2)

Figura (es. n. 1)

stampa del valore del record considerato.

Un END conclude la routine (nel caso di subroutine, si sostituirà END con RETURN). Nelle righe 140-160 possiamo notare la presenza dei DATA che, come prima spiegato, possono essere ampliati e digitati a casaccio dal lettore.

Il secondo listato

Passando ora alla spiegazione del programma Basic N.2 cominceremo ad analizzare la riga 80, in quanto le precedenti sono già state analizzate ampiamente.
Dopo il solito ciclo di riempimento del vettore e di azzeramento di TI$, troviamo l'assegnazione del valore 2 alla variabile C. L'utilizzo di tale variabile, si è reso necessario per decidere se utilizzare o meno, il record ennesimo. Questo lo possiamo notare alla successiva riga 90, nella quale è presente l'IF necessario a controllare se il vettore è dimensionato con un numero pari di records o meno: se si verificasse la prima ipotesi (N pari), C non varierà e di conseguenza tutti i records formanti il vettore reale saranno presi in considerazione. In caso contrario, l'ennesimo record verrebbe riempito con un valore altissimo e preso in considerazione ma mai spostato dalla sua posizione (C=1 permette di fare ciò).

Alla riga 110, troviamo il ciclo primario di I che andrà da 0 a N-C, con un avanzamento di due per i motivi spiegati nell'analisi dei FLOW-CHART. Di seguito notiamo l'assegnazione a B del valore di partenza I+1, seguito dal ciclo secondario J, che varierà da B (=I+1 nella routine precedente) a N-1.

Alla riga 120, troviamo l'IF di controllo del valore minore. Nel caso in cui A(I) risultasse maggiore di A(J), avremmo l'aggiornamento della variabile B che verrà posta uguale a J, seguito dallo scambio dei valori dei record, ancora grazie all'utilizzo di H come variabile intermedia (l'avevamo trovata nella routine precedente). L'unica istruzione presente alla riga 140, è il NEXT di fine del ciclo J. Nella successiva riga 150, è presente l'assegnazione di B e D, seguita dall'inizio del ciclo terziario T di ricerca del valore da inserire nel record I+1. Un IF confronta il valore di A(D) e A(T): nel caso in cui il primo risultasse minore del secondo non si avrebbero variazioni. In caso contrario la variabile D verrebbe aggiornata (posta uguale a T).

Alla riga 160, troviamo il NEXT di chiusura del ciclo di T e il successivo scambio di valori tra i records I+1 e D, utilizzando ancora H. A questo punto (riga 170), troviamo la chiusura del ciclo I, seguita dalla stampa del tempo impiegato per svolgere il processo e del vettore ormai ordinato. Nel gruppo 180-210, troviamo i DATA.

Luca Cirillo

```
10 REM ***************************
20 REM * ORDINAMENTO RAPIDO DI    *
30 REM * UN VETTORE  DI NUMERI    *
40 REM *                          *
50 REM *      LUCA CIRILLO        *
60 REM ***************************
61 :
70 CLR:INPUT"RECORDS ";N:DIMA(N)
80 FORI=1TON-1:READA(I):NEXT:TI$="000000":PRINTTI$:C=2
90 IF N<>INT(N/2)*2THEN110
100 A(N)=10↑38:C=1
110 FORI=0TON-C STEP2:B=I+1:FORJ=BTON-1:
120 IF A(I)<=A(J)THEN140
130 B=J:H=A(I):A(I)=A(J):A(J)=H
140 NEXTJ
150 D=B:FORT=BTON-1: IFA(D)>A(T)THEND=T
160 NEXTT:H=A(I+1):A(I+1)=A(D):A(D)=H
170 NEXTI:PRINTTI$:PRINT"PREMI UN TASTO"
171 GETA$: IFA$=""THEN171
175 FORI=0TON-1:PRINTA(I),I:NEXTI:END
180 DATA 25,18,-34,123,0,0,234,-45,2E-21
185 DATA 1E-3,45,-23,56,23,0.456,-,67,765
190 DATA -34.56,-56,77,111,34,45,0,-56
195 DATA 218,-34,518,-64.6,-64,5,54,82,-56
200 DATA -334,78.456,678,679,600,681,682
205 DATA 66,-11.45,-76,54,.876,-1,9,-1200,2,4
210 DATA 158,152,12,85,86,75,980,1.2345
215 DATA 42,7,6,146,67,155,2,8,9,6,5,4,3,2,1,0
READY.
```

```
100 REM ******************************
110 REM * ORDINAMENTO DI UN VETTORE *
120 REM *                            *
130 REM *       LUCA CIRILLO         *
140 REM ******************************
150 :
160 CLR:INPUT"■■■■ N. RECORDS ";N:DIM IV(N)
170 FORI=1TON:READ IV(I):NEXTI:TI$="000000":PRINT"■"TI$
180 FORI=1TON-1:FORJ=I+1TON
190 IFIV(I)<IV(J)THEN210
200 H=IV(I):IV(I)=IV(J):IV(J)=H
210 NEXT J,I:PRINTTI$
220 PRINT"PREMI UN TASTO"
230 GETA$:IFA$=""THEN230
240 FORI=1TON:PRINT"N RECORD "I;
250 PRINTSPC(4)"VALORE REC. "IV(I):NEXT I:END
260 DATA 25,18,-34,123,0,0,234,-45,2E-21
270 DATA 1E-3,45,-23,56,23,0.456,-,67,765
280 DATA -34.56,-56,77,111,34,45,0,-56
290 DATA 218,-34,518,-64.6,-64,5,54,82,-56
300 DATA -334,78.456,678,679,600,681
310 DATA 682,66,-11.45,-76,54,0.876,-1

READY.
```

DIGITARE STANCA

I programmi più interessanti spesso sono molto lunghi, un listato pubblicato è faticoso da leggere...

Commodore Computer Club vi offre un'alternativa: le cassette con tutti i programmi pubblicati sulla rivista.

Ogni nastro contiene il software di un numero di Commodore Computer Club a un prezzo incredibilmente basso: solo 5.800 lire (+ 1.000 lire per spese di spedizione).

Riceverete le cassette direttamente a casa vostra, utilizzando il coupon qui a fianco.

Desidero ricevere le cassette con il software pubblicato sui seguenti numeri di Commodore Computer Club:

....

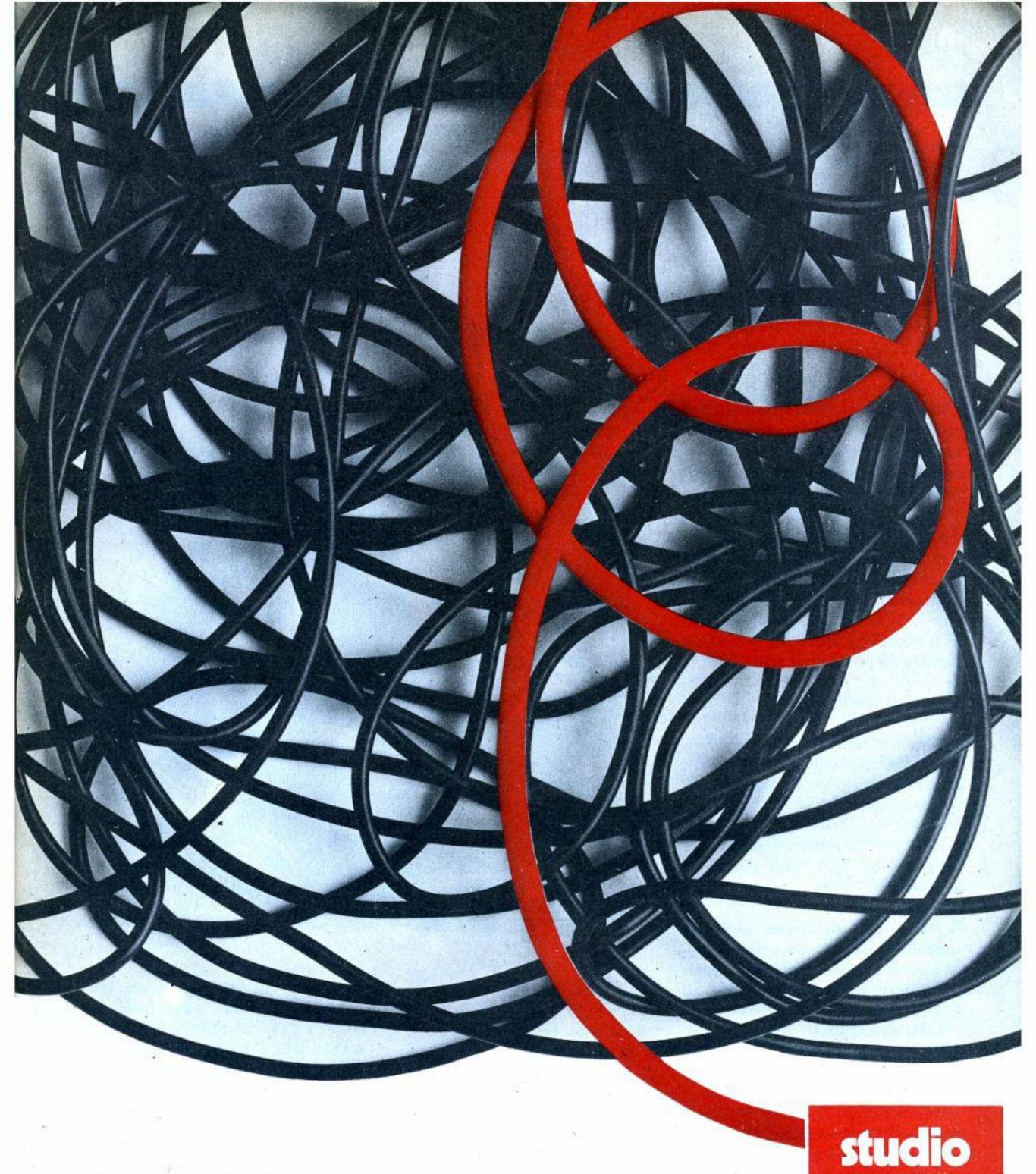
importo L. ....
spese di spedizione L. 1.000

Totale L. ....

☐ ho versato l'importo sul c/c postale n. 31532203 (allego fotocopia della ricevuta di versamento)
☐ accludo assegno non trasf. n. ....
(banca ....)
intestato a C.C.C, viale Famagosta, 75-20142 Milano

nome ....
cognome ....
via ....
CAP/città ....

Ritagliare e spedire in busta chiusa a: Systems Editoriale v.le Famagosta 75, 20142 Milano.

Abbonatevi a Commodore Computer Club

GRAVITAZIONE E SATELLITI

Questo programma permette di simulare sul Commodore 64 il comportamento di un satellite orbitante attorno alla Terra o a qualunque altro pianeta.

Sono qui presentate due versioni dello stesso programma: la prima permette di visualizzare l'orbita descritta dal satellite direttamente sullo schermo sfruttando le nuove routines grafiche di D. Toma, mentre la seconda consente di averne una copia cartacea con la stampante plotter 1520.

Un po' di teoria

Come tutti saprete, la forza che si esercita tra due corpi di massa "M" ed "m", secondo la legge di gravitazione universale, vale in modulo:

$F= -G*M*m/(r\uparrow 2)$ [1]

in cui "G" è la costante di gravitazione universale "M" ed "m" sono le masse dei due corpi ed "r" è la distanza tra essi.

Questa formula diventa molto più comprensibile se espressa in forma vettoriale, in quanto oltre a indicare il modulo della forza ne indica anche la direzione e il verso.

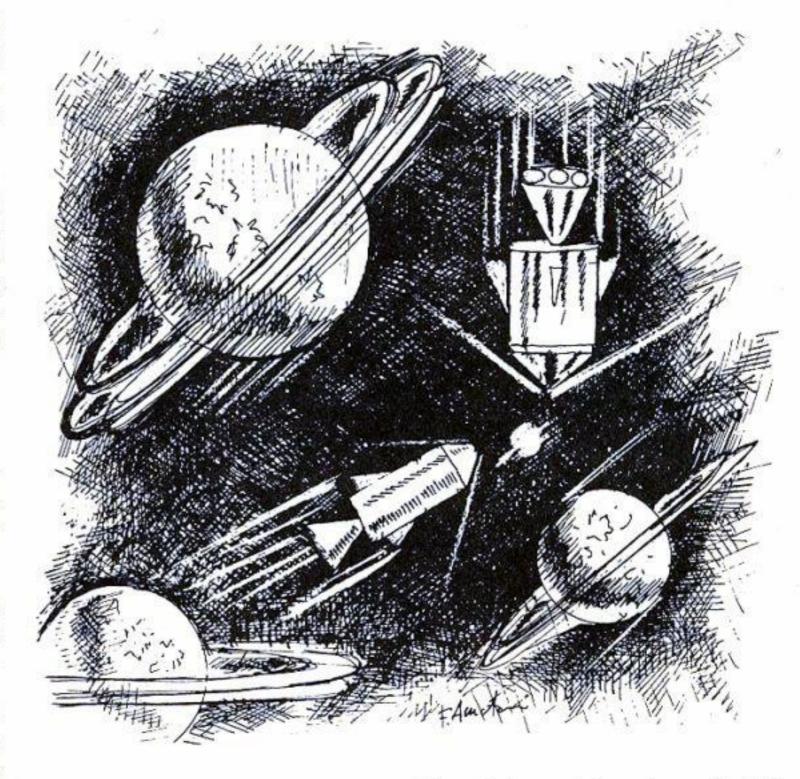
La [1] diventa quindi:

$\vec{F}= (-G*M*m/(r\uparrow 2)) *\vec{r}/r$ [2]

Il termine "$\vec{r}/r$" viene chiamato VERSORE, in quanto valendo "1" in modulo e moltiplicato per la [1] non ne influenza il valore ma in compenso indica la direzione e il verso della forza gravitazionale.

Con un semplice passaggio matematico la [2] può essere riscritta:

$\vec{F}= (-G*M*m/(r\uparrow 3)) * \vec{r}$ [3]

Dato che, nel caso particolare, interessa calcolare l'orbita percorsa dal satellite attorno al centro di massa del sistema gravitazionale, possiamo applicare la ben nota legge fondamentale della dinamica:

$\vec{F}= m*\vec{a}$ [4]

Ricordando, ovviamente, che:

$\vec{a}= d\vec{v}/dt$.

la [4] può così essere scritta:

$\vec{F}= m*dv/dt$ [5]

Unendo ora la [3] con la [5] e ricordando che:

$\vec{v}= d\vec{s}/dt$

in cui "s" rappresenta lo spazio percorso dal satellite, otteniamo il sistema:

$d\vec{v}\ /dt= (G*M/(r\uparrow 3))*r$

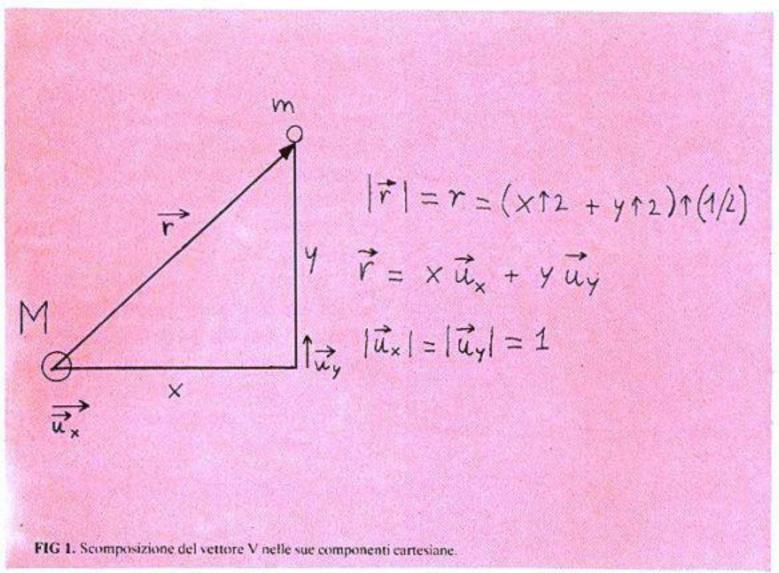
$v= d\vec{s}/dt$

Dalla prima equazione si può trarre l'importante conclusione che, grazie al fatto che "m" è stato semplificato, LA FORZA CHE SI ESERCITA SU UN SATELLITE È INDIPENDENTE DALLA SUA MASSA e quindi anche l'orbita da esso percorsa.

Questo spiega come mai gli astronauti non abbiano bisogno di rimanere agganciati all'astronave madre quando escono nello spazio. Dato che interessa rappresentare l'orbita bidimensionale su un piano cartesiano, dovremo scomporre tutti i vettori nelle loro componenti cartesiane (vedi fig. 1).

Il sistema potrà quindi essere riscritto:

$d\vec{v}/dt = -G*M*(X*\vec{U}x+Y*Uy)/$

FIG 1. Scomposizione del vettore V nelle sue componenti cartesiane.

(X↑2+Y↑2)↑3/2
$\vec{v} = d\vec{s}/dt$

Anche i vettori "v" e "s" possono essere scomposti secondo le loro componenti cartesiane (fig. 2) ottenendo così:

dVx/dt = -G*M*X/(X↑2+Y↑2)↑3/2
dVy/dt = -G*M*Y/(X↑2+Y↑2)↑3/2
dX/dt = Vx
dY/dt = Vy

Da tale sistema si nota, purtroppo, che per conoscere le coordinate x ed y del nostro satellite è necessario conoscerne la velocità lungo gli assi x ed y.

D'altra parte, per conoscere la velocità è necessario conoscere le coordinate x e y del satellite stesso.

La soluzione di questo circolo vizioso comporterebbe la risoluzione di una equazione differenziale del secondo ordine, problema matematico di estrema difficoltà, anche per un computer.

Fortunatamente si può aggirare l'ostacolo con un semplice trucco: se invece di calcolare la velocità istantanea del satellite per ogni intervallo di tempo infinitesimale "dt" poniamo:

dt = E

in cui "E", espresso in secondi, rappresenta il tempo di campionamento, potremo scrivere:

a) Vx=(-G*M*X/(X↑2+Y↑2)↑3/2)*E [6]
b) Vy=(G-*M*Y/(X↑2+Y↑2)↑3/2*E [7]
c) X=Vx*E [8]
d) Y=Vy*E [9]
e) E=E+Ei (Ei=tempo iniziale) [10]

In conclusione:

- Poste le condizioni iniziali: X=Xi, Y=Yi, Vx=Vxi, Vy=Vyi, E=Ei possiamo tramite la [6] e la [7] calcolare le nuove "Vx" e "Vy".
- Tramite la [8] e la [9] possiamo calcolare le nuove e tanto sospirate coordinate X e Y.
- Infine (era ora, diranno i due o tre lettori giunti fin qui), dopo aver eseguito la [10] possiamo tornare al punto a) e continuare così fino a quando abbiamo calcolato tutte le coppie X, Y di cui avevamo bisogno per disegnare l'orbita.

Come girano i due programmi

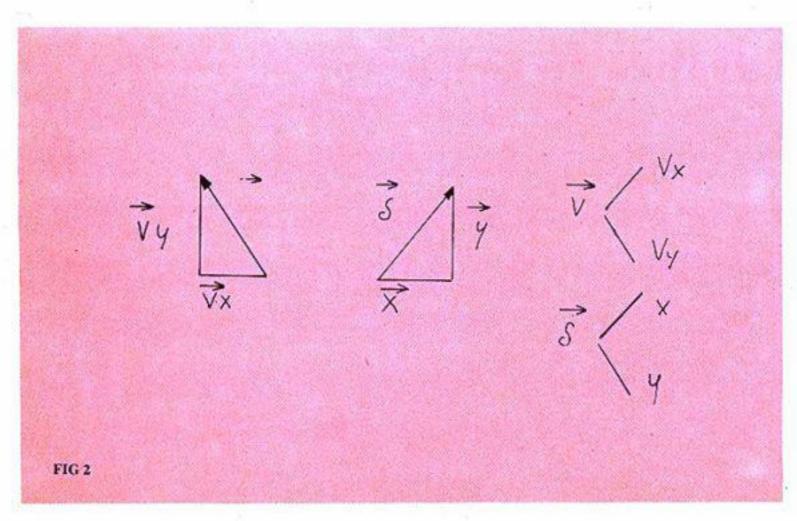
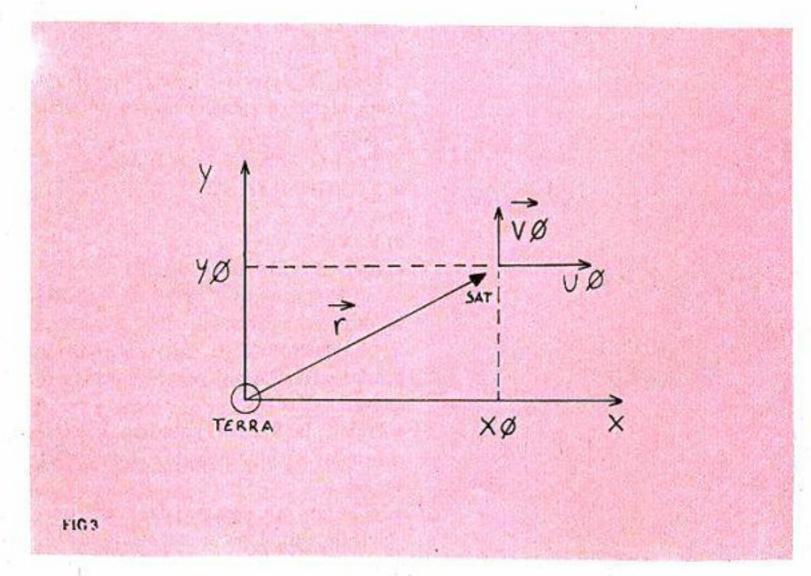
Il programma presentato nelle due versioni, non fa altro che applicare i semplici (!) calcoli visti per disegnare il percorso di un satellite orbitante attorno alla terra (linea 91). Prima di dare il RUN ricordatevi di usare il programma giusto e di caricare e lanciare PRIMA le nuove routines di D. Toma (C.C.C. N. 14) se volete un grafico sullo schermo.

Il computer, non appena "parte" il programma, chiederà di introdurre i dati iniziali X0, Y0 che sono le coordinate del satellite rispetto ad un sistema di assi cartesiani ortogonali che hanno come origine il centro della terra, e che vanno espressi in metri.

Chiederà, inoltre, V0 e U0, componenti della velocità iniziale rispettivamente sull'asse delle X e su quello delle Y, e che vanno espresse in metri al secondo (m/s). STEP non è altro che "Ei" visto prima e che determina l'intervallo di campionamento delle coordinate e, in pratica, la risoluzione del disegno. Va espresso in secondi. Se non avete capito quasi nulla provate a guardare la figura 3.

A questo punto, dato il RETURN, il

FIG 2

FIG 3

computer inizia a tracciare il disegno dell'orbita.

È da notare che il calcolatore entra in un loop indefinito e, di conseguenza, occorrerà fermare "a mano" il programma una volta terminato il tracciamento dell'orbita premendo i tasto STOP (versione PLOTTER) o i tasti STOP e RESTORE (versione video).

Se fermate il programma esattamente nel momento in cui il satellite sta per iniziare la seconda orbita, potete, eseguendo il comando diretto PRINT T, visualizzare il tempo impiegato (espresso in secondi) per percorrere un'orbita.

Esempio pratico

Dato il RUN, introducete i seguenti da-

```
1 REM          GRAVITAZIONE UNIVERSALE.
2 REM  VERSIONE VIDEO PER COMMODORE 64 & ROUTINE TOMA.
3 REM       ANDREA BORIANI & GIUSEPPE CIANI
4 GOSUB2000
5 ←CLEAR:←GRAF0,1:←COLOR1:←CIRCLE0,0,0,5,5
10 G=6.67
91 M=5.97E13
110 F=(-G*M*X)/((X↑2+Y↑2)↑(3/2))
115 D=(-G*M*Y)/((X↑2+Y↑2)↑(3/2))
120 V=V+F*(E/2)
130 U=U+D*(E/2)
140 GOTO180
150 REM **** INIZIO CICLO DI CALCOLO ****
151 F=(-G*M*X)/((X↑2+Y↑2)↑(3/2))
152 D=(-G*M*Y)/((X↑2+Y↑2)↑(3/2))
160 V=V+F*E
170 U=U+D*E
180 X=X+V*E
190 Y=Y+U*E
195 T=T+1:IFT=1THEN←PLOTX/4000000,Y/4000000,0:GOTO150
197 ←PLOTX/4000000,Y/4000000,0
210 GOTO150
2000 PRINT"ESEMPIO:3.84E8, 0, 0, 1022.3488, 3600"
2005 PRINT"INSERISCI LE CONDIZIONI INIZIALI (X0,Y0,V0,U0,STEP)"
2010 INPUTX,Y,V,U,E
2020 PRINTX;",";Y;",";V;",";U;",";E
2030 RETURN
READY.
```

ti:
3.84E8. 0. 0. 1022.3488. 3600

Il satellite in questione dista quindi dalla terra 384000 km, gli è stata data una velocità iniziale di circa 1022 m/s, presenta fasi ben visibili, risveglia i lupi mannari... Insomma, avrete capito che si tratta della nostra Luna!

Dato il RETURN verrà tracciata l'orbita del satellite naturale della Terra. Se fermate il programma non appena sta per iniziare la seconda orbita, digitate:
PRINT T/(3600*24)

Vedrete che verrà visualizzato il numero 27 che rappresenta, appunto, il periodo di rotazione attorno alla Terra in giorni.

Provate ora a vedere cosa succederebbe se alla Luna fosse impressa una velocità iniziale "U0" di 1200 m/s o di 400 m/s o di 1400 m/s. Per concludere un ultimo avviso: se il disegno dell'orbita risultasse troppo grande o troppo piccolo per essere apprezzato, dovrete modificare il fattore di scala che appare nelle linee 195 e 197 (il numero 4000000, quattro milioni). Questo numero dovrà essere aumentato per rimpicciolire il disegno e viceversa.

A proposito, l'orbita risulterà sempre un'ellissi, tranne nei casi in cui la velocità iniziale sia superiore alla velocità di fuga dal sistema, per cui il satellite si allontana con un modo parabolico o iperbolico a seconda della sua velocità.

Alberto & Andrea Boriani

```
1 OPEN2,6,0
4 GOSUB2000
5 OPEN1,6,1:PRINT#1,"M",220,-200
10 G=6.67
60 PRINT#1,"I"
61 FORQ=0TO2*π STEP.05
62 PRINT#1,"J",10*SIN(Q);10*COS(Q)
63 NEXT
91 M=5.97E13
110 F=(-G*M*X)/((X↑2+Y↑2)↑(3/2))
115 D=(-G*M*Y)/((X↑2+Y↑2)↑(3/2))
120 V=V+F*(E/2)
130 U=U+D*(E/2)
140 GOTO180
150 REM **** INIZIO CICLO DI CALCOLO ****
151 F=(-G*M*X)/((X↑2+Y↑2)↑(3/2))
152 D=(-G*M*Y)/((X↑2+Y↑2)↑(3/2))
160 V=V+F*E
170 U=U+D*E
180 X=X+V*E
190 Y=Y+U*E
195 T=T+1:IFT=1THENPRINT#1,"R";X/4000000;Y/4000000:GOTO150
197 PRINT#1,"J";X/4000000;Y/4000000
210 GOTO150
2000 PRINT#2,"CONDIZIONI INIZIALI (X0,Y0,V0,U0,STEP)"
2010 INPUTX,Y,V,U,E
2020 PRINT#2,X;",";Y;",";V;",";U;",";E
2030 RETURN
2899 REM ****************************
2900 REM   GRAVITAZIONE UNIVERSALE.
2901 REM       VERSIONE PLOTTER
3000 REM
3010 REM      DI ANDREA BORIANI
3020 REM              E
3030 REM        GIUSEPPE CIANI
3050 REM ****************************
READY.
```

BORDO DEL VIDEO IN TECHNICOLOR

Come sfruttare l'interrupt per creare effetti inconsueti come la colorazione di numerose strisce sul bordo del televisore.

Per fare un piccolo esempio dell'utilizzo del RASTER REGISTER, basta citare il bellissimo effetto di prospettiva che si trova nei giochini più sofisticati per il COMMODORE 64.

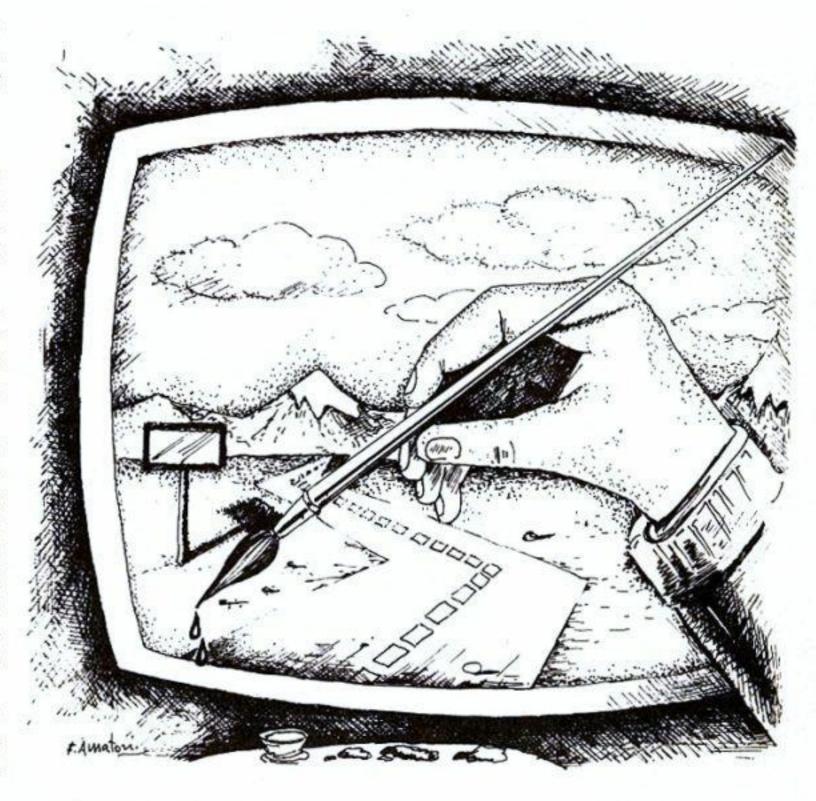
L'effetto, ad esempio, delle nuvole, montagne e strada che si muovono a velocità diverse sullo schermo per dare il senso di profondità, si realizza appunto solo con l'ausilio del RASTER REGISTER.

Un esempio di utilizzo dell'interrupt è dato dal COMMODORE 64 stesso. Il vostro computer infatti funziona completamente in interrupt. La scansione della tastiera ad esempio avviene in interrupt, e quando premete i tasti RUN/STOP e RESTORE generate un interrupt di tipo particolare che il vostro 64 non può assolutamente ignorare. Consiglio di rileggere l'articolo pubblicato sul n. 13 (interrupt).

L'interrupt è dunque un segnale elettrico che induce il microprocessore a compiere una precisa operazione, per l'esattezza un salto ad una routine stabilita.

Esistono due tipi fondamentali di interrupt: quello mascherabile e quello non mascherabile.

Il primo può essere abilitato o disabilitato da programma con le istruzioni assembler CLI e SEI (rispettivamente per l'abilitazione e la disabilitazione). Ciò significa che, volendo, si può ignorare una richiesta di interrupt mascherabile.

Il secondo tipo di interrupt, invece, deve essere comunque riconosiuto dalla macchina ed è appunto il tipo di segnale generato dalla pressione dei tasti RUN/STOP e RESTORE.

Quando il COMMODORE 64 riceve una richiesta di interrupt, cede immediatamente il controllo delle operazioni alla routine che si trova alla locazione di memoria specificata dal contenuto delle locazioni 0314 e 0315 (esadecimali) che normalmente "puntano" ad ER31 (esadecimale) dove si trova la routine che scandisce la tastiera e gestisce gli interrupt.

Se provate a modificare queste due locazioni di memoria (0314-0315) ponendovi l'indirizzo di una vostra routine in linghaggio macchina, otterrete l'esecuzione continua della vostra routine a cui il computer salterà ogni volta che riceve un interrupt.

Leggendo il registro 18 del VIC sappiamo quindi, istante per istante, dove il computer è arrivato a "pennellare" l'immagine video. Potremmo quindi fare un piccolo programmino in BASIC che controlli quando il RASTER REGISTER è arrivato a 151 (ossia a metà schermo) e che a quel punto cambi il colore di sfondo e lo ricambi quando il RASTER REGISTER è arrivato a 0. In questo modo otterremo sullo schermo due colori di sfondo diversi ad esempio il bianco per la parte superiore ed il nero per la parte inferiore.

Ci accorgeremo ben presto però che

questo programmino non funziona a causa della lentezza del BASIC. Il RASTER REGISTER, infatti, viene incrementato ad una velocità tale che, se il nostro prommamino comincia l'operazione di cambiamento del colore di sfondo quando il RASTER REGISTER è arrivato al valore 151, conclude la stessa operazione quando ormai il RASTER REGISTER ha oltrepassato il valore 200.

Il nostro programma in BASIC sarebbe quindi troppo "lento di riflessi" per ottenere una divisione netta sullo schermo. Arriviamo quindi a capire l'utilità del secondo registro che si trova sempre a D012. Questo secondo registro è a sola scrittura, a differenza del primo che era a sola lettura, ossia vi si può accedere con una POKE ma non si riesce a rileggerlo con una PEEK. Rileggendo la D012 infatti non troveremmo il numero che abbiamo "POKato" ma bensì il valore del contatore di quadro come abbiamo detto sopra. Questo secondo registro dice al VIC di inviare un segnale di interrupt quando il contatore di quadro è arrivato al valore che abbiamo inserito.

Ecco dunque risolti tutti i nostri problemi; ci basta mettere 151 nel registro 18, aspettare che arrivi un interrupt dal RASTER REGISTER e cambiare il colore di sfondo. Quando arriva l'interrupt, infatti, noi siamo sicuri che il RASTER REGISTER è arrivato a 151, ossia a metà schermo, e questa certezza l'abbiamo raggiunta senza bisogno di leggere continuamente il RASTER REGISTER per sapere dove fosse arrivato.

Ecco come nasce la routine che presento con questo articolo: basta andare a modificare i vettori di interrupt 0314 e 0315, come abbiamo visto prima, in modo che ogni volta che arriva un interrupt il computer salti a C022 (esadecimale). A questo indirizzo piazzeremo una seconda routine che ha il compito di controllare se l'interrupt ci giunge effettivamente dal RASTER REGISTER, se così è, cambiamo il colore del bordo dello schermo ogni 16 righe in modo da ottenere tante strisce colorate e, una volta arrivat in fondo allo schermo, passiamo il controllo alla routine che si occupa di scandire la tastiera e che si trova ad EA31 (esadecimale).

La vostra routine dovrà terminare con un salto ad EA31 in modo che poi, una volta eseguita, il computer possa continuare a gestire i suoi "interrupts".
In questo modo, se ponete in 0314 e 0315 l'indirizzo di una routine che, ad esempio, suona una musichetta, vi accorgerete che il computer continua a funzionare regolarmente, ossia potete scrivere un programma in BASIC, mentre la musichetta viene suonata di continuo. In pratica, il computer eseguirà due programmi contemporaneamente: quello che suona la musichetta e un qualsiasi programma in BASIC che volete far girare.

State attenti a non modificare il contenuto delle 0314 e 0315 prima di aver disabilitato l'interrupt, altrimenti il computer si inchioderà per un semplice motivo: se il microprocessore riceve un interrupt mentre state modificando i puntatori eseguirà un salto a caso nella memoria e continuerà ad eseguire lo stesso salto ad ogni interrupt ricevuto senza che possiate terminare di modificare i puntatori.

È preferibile quindi disabilitare l'interrupt prima della modifica e riabilitarlo subito dopo. Per fare questo potete scrivere 75 (esadecimale) nel registro di controllo delle interruzioni della prima CIA, il quale si trova a DC0D (esadecimale), modificare i vettori 0314 e 0315 inserendovi l'indirizzo della vostra routine (prima la parte bassa poi quella alta) ed infine riabilitare l'interrupt mettendo 81 (esadecimale) in DC0D; questa routine poi terminerà con un RTS in modo da tornare al BASIC.
Attenzione! Questa routine deve essere lanciata quando la routine che volete far lavorare in iterrupt già esiste, altrimenti il computer salterà comunque a quell'indirizzo una volta ricevuto il primo interrupt, ma non troverà la vostra routine.

La routine che parte da C000 (esadecimale) o, se preferite, da 49152 (decimale) funziona come detto sopra ed inoltre compie altre due operazioni: abilita gli interrupts generati dal RASTER REGISTER mettendo 1 in D01A (esadecimale) ossia nel registro 26 del VIC ed inoltre azzera il bit più significativo del RASTER REGISTER.

Vediamo ora che cosa intendiamo quando parliamo di RASTER REGISTER.
Il registro 18 del VIC o, il che è lo stesso, la locazione D012 contiene in realtà due registri ben distinti che convivono in modo particolare. Per l'esattezza, quando leggiamo questa locazione (ad esempio con una PEEK), accediamo ad un particolare contatore che ci restituisce la posizione corrente del "pennellino" elettronico che scandisce lo schermo video. Più semplicemente se troviamo 0 in questo registro vuol dire che il computer sta trasmettendo al televisore i dati della prima riga (che si trova addirittura fuori dallo schermo visibile); se invece troviamo 51, vuol dire che in questo momento sullo schermo arrivano i segnali corrispondenti alla prima riga visibile, se invece troviamo 251 significa che sullo schermo arrivano i segnali corrispondenti all'ultima riga visibile e così via. Ricordiamo che per "riga" si intende quella tracciata dal cannone elettronico.

Passiamo ora ad analizzare più in dettaglio il disassemblamento della routine.
Dalla C000 alla C011 vengono modificati i vettori di interrupt come spiegato sopra ossia disabilitando l'interrupt, modificando i vettori 0314 e 0315 e riabilitando l'interrupt.

Le istruzioni C014 e C016 si occupano invece di mettere 1 nel registro 26 del VIC: in questo modo noi diciamo al VIC di inviare un interrupt ogni qual volta il valore del RASTER REGISTER corrisponde al valore che noi metteremo nel registro 18 (D012 in esadecimale).

Dalla C019 ala C01E cancelliamo il bit più significativo del registro 17.
Il RASTER REGISTER (ossia il contatore di quadro ed il registro di confronto assieme a D012) non è un registro ad otto bit, bensì a nove, e visto che a noi interessano solo i valori da 0 a 255 cancelliamo il nono bit che si trova appunto nel registro 17. È infatti errato dire che il RASTER REGISTER si trova all'indirizzo D012: qui infatti si trovano solo gli otto bit più bassi del RASTER REGISTER mentre il nono ed ultimo bit si trova a D011 ed esattamente l'istruzione C021 provoca il ritorno al BASIC.

Questa routine infatti, come abbiamo

```
RAS......PAGE 0001

LINE# LOC   CODE          LINE

00001 0000                *=49152
00002 C000  A9 7F              LDA #$7F
00003 C002  8D 0D DC           STA $DC0D      ; DISABILITA L'INTERRUPT
00004 C005  A9 22              LDA #$22       ; MODIFICA IL VETTORE INTERRUPT
00005 C007  8D 14 03           STA $0314      ; E VI PONE L'INDIRIZZO
00006 C00A  A9 C0              LDA #$C0       ; DELLA NOSTRA ROUTINE
00007 C00C  8D 15 03           STA $0315
00008 C00F  A9 81              LDA #$81       ; RI-ABILITA L'INTERRUPT
00009 C011  8D 0D DC           STA $DC0D
00010 C014  A9 01              LDA #$01       ; ABILITA L'INTERUPT DEL
00011 C016  8D 1A D0           STA $D01A      ; RASTER REGISTER
00012 C019  AD 11 D0           LDA $D011      ; AZZERA IL BIT
00013 C01C  29 7F              AND #$7F       ; PIU' SIGNIFICATIVO
00014 C01E  8D 11 D0           STA $D011      ; DEL RASTER REGISTER
00015 C021  60                 RTS            ; RITORNA AL BASIC
00016 C022  AD 19 D0           LDA $D019      ; CONTROLLA DA DOVE VIENE
00017 C025  29 01              AND #$01       ; L'INTERRUPT E SE ARRIVA DAL
00018 C027  D0 03              BNE $C02C      ; RASTER REGISTER CONTINUA
00019 C029  4C BC FE           JMP $FEBC      ; ALTRIMENTI TORNA DALL'INTERRUP
T
00020 C02C  8D 19 D0           STA $D019      ; SCRIVE NEL REGISTRO INDICATORE
00021 C02F  AD 00 C8           LDA $C800      ; DI INTERRUZIONE,LEGGE UN COLOR
E
00022 C032  8D 20 D0           STA $D020      ; E LO PONE NEL REGISTRO DI BORD
O
00023 C035  EE 00 C8           INC $C800      ; INCREMENTA IL COLORE
00024 C038  AD 00 C7           LDA $C700      ; LEGGE UN NUMERO QUALSIASI
00025 C03B  18                 CLC            ; DA UNA LOCAZIONE DI MEMORIA
00026 C03C  69 10              ADC #$10       ; VI SOMMA DIECI E LO RIPONE
00027 C03E  8D 00 C7           STA $C700      ; NELLA STESSA LOCAZIONE
00028 C041  8D 12 D0           STA $D012      ; E ANCHE NEL RASTER REGISTER
00029 C044  90 E3              BCC $C029      ; SE E' PARI A 255 VA A GESTIRE
00030 C046  4C 31 EA           JMP $EA31      ; LA TASTIERA ALTRIMENTI RITORNA
```

detto prima, funziona in modo trasparente rispetto al BASIC, possiamo cioè tranquillamente eseguire, scrivere o caricare un programma mentre i colori del bordo vengono cambiati.

Con questa prima routine abbiamo quindi fatto in modo che ogni qual volta viene generato un interrupt il controllo passi alla routine che comincia a C022, abbiamo inoltre attivato l'interrupt proveniente dal RASTER REGISTER ed abbiamo cancellato il bit più significativo dello stesso.

Le istruzioni dalla C022 alla C029 svolgono questa funzione: controllano il bit meno significativo del registro 25 (D019 esadecimale) il quale ci dice se l'interrupt che è appena giunto viene dal RASTER REGISTER o no. Se l'interrupt non giunge dal RASTER REGISTER salta a FEBC (esadecimale), ossia esce dalla routine e passa il controllo al BASIC, se invece l'interrupt viene proprio dal RASTER REGISTER continua la routine.

La C02C scrive 1 nel registro 25; anche il registro 25 infatti è un doppio registro e se noi lo leggessimo senza riscriverci 1 non otterremmo nessun altro segnale di interrupt dal RASTER REGISTER.

Dalla C02F alla C032 il programma prende il valore contenuto nella locazione di memoria C800 e lo mette nel registro 32 (D020), ossia il registro che contiene il colore di bordo dello schermo.

La C035 incrementa la locazione che contiene il colore di bordo dello schermo.

Dalla C038 alla C03E viene preso il numero dalla locazione C700 vi si somma 10 in esadecimale (ossia 16 in decimale) e lo si ripone di nuovo in C700.

La C041 si occupa di mettere di volta in volta questo numero, prelevato dalla locazione C700, nel RASTER REGISTER, in modo da ottenere una richiesta di interrupt ogni 16 linee di schermo. Il risultato finale sarà quello di ottenere un cambiamento del colore di bordo ogni 16 linee.

La C044 controlla se il RASTER REGISTER è arrivato in fondo allo schermo. Se così è stato passa il controllo alla routi-

ne di gestione della tastiera, la quale altrimenti rimarrebbe insensibile al nostro tocco! Oppure restituisce il controllo al sistema operativo in attesa del prossimo interrupt che arriverà puntualmente tra 16 linee di schermo.
Attenzione: per 16 linee di schermo, ripeto, non intendo 16 linee di caratteri ma bensì 16 linee elementari pari a circa due linee di caratteri.

La spiegazione dettagliata della routine non dovrebbe lasciare dubbi sul suo principio di funzionamento e vi sarete sicuramente resi conto che quello che vi propongo è solo un piccol esempio della straordinaria potenza del RASTER REGISTER. Provate a digitare, mentre il bordo cambia colore, POKE 49213, I in cui la variabile "I" rappresenti una potenza di 2.
Fate quindi attenzione ad inserire correttamente le poche linee di DATA e se non commetterete errori ne vedrete proprio di tutti i colori!

Luca Galuzzi

```
100 REM ****************************
110 REM  R A S T E R : APPLICAZIONI
115 REM *                          *
120 REM *      DI LUCA GALUZZI     *
130 REM *  SOLO PER COMMODORE 64   *
150 REM ****************************
490 CLR : RESTORE
500 FOR X = 49152 TO 49224
510 READ A
520 POKE X , A
530 NEXT
540 SYS 49152
1000 DATA169,127,141,13,220,169,34,141,20,3
1010 DATA169,192,141,21,3,169,129,141,13,220
1020 DATA169,1,141,26,208,173,17,208,41,127
1025 DATA141,17,208,96,173,25,208,41,1,208
1030 DATA3,76,188,254,141,25,208,173,0,200
1040 DATA141,32,208,238,0.200,173,0,199,24
1050 DATA105,16,141,0,199,141,18,208,144,227
1060 DATA76,49,234
READY.
```

COESISTENZA DI PIU' PROGRAMMI BASIC

Come allocare in memoria RAM due o più programmi Basic selezionabili mediante semplici istruzioni POKE.

In questo articolo viene descritta una tecnica, utile in più di un caso, che consente di "segmentare" a piacere la memoria a disposizione dell'utente.

Indispensabile si rivelerà lo studio dell'articolo "Un pò d'ordine tra i bit" (n. 10 di C.C.C.) per coloro che vorranno applicare su larga scala le nozioni del presente articolo.
La semplice, ma comoda, applicazione riportata in queste pagine non è altro, infatti, che una tra le mille che il lettore potrà utilizzare a proprio vantaggio.

Il canale di errore del drive 1541

Quando ci accingiamo, per la prima volta, a gestire files su disco, può capitare, nel leggere o scrivere records, che si blocchi il computer oppure che l'unità dischi si fermi mentre il led rosso si accende nel caso si tratti di un errore di I/0 (Input/Output) o di logica.

Per ovviare a questo inconveniente sarà utile battere "RUN STOP" e "RESTORE" tornando in tal modo all'EDITOR e al prompt di sistema (READY). Il problema, comunque, sussiste in quanto non possiamo sapere se, dando ancora il "RUN", il computer si inchioderà nuovamente.

Lo scopo della routine pubblicata è quello di fornire, all'utente, i dati necessari affinché si renda conto di quale errore è stato commesso nella gestione dei files e, di conseguenza, modificare il programma in modo che non sussista più l'errore rilevato.

Nella memoria centrale del drive, sono contenuti i tipi di errore possibili, circa 74, ognuno dei quali è individuato da un numero (ERRORE N), una superficialissima spiegazione (TIPO DI ERRORE), il numero della traccia (TRACK) e del blocco (BLOCK), nel quale si è verificato.

La routine apre un canale, rileva il numero dell'errore (se sussiste), il tipo di errore, la traccia e il blocco nei quali si è verificato e li visualizza sullo schermo.

Una volta attivata, compariranno sullo schermo quattro messaggi:

Il primo rappresenta il numero dell'errore (compreso tra zero e 74)

Il secondo è il tipo di errore seguito dalla descrizione dell'errore (es. FILE NOT FOUND, DRIVE NOT READY, ecc.);

Il terzo è il numero della traccia (valore compreso tra 0 e 35) in cui si è verificato l'eventuale errore.

Il quarto messaggio rappresenta il numero del blocco compreso tra 0 a 664. In realtà i blocchi del disco sono 683 ma 18 sono riservati alla Directory.

Come utilizzare la routine

Per utilizzare correttamente la routine è necessario:

tine in quanto è allocata dalla locazione 2049 in poi. Stesso risultato avremo con qualsiasi altro comando BASIC che non intacchi i bytes d'inizio della memoria. Anche se proviamo a caricare un programma basic avremo lo stesso risultato.

Può venire spontaneo chiedere come riattivare la routine senza doverla ricaricare. Ebbene, la risposta è semplicissima: basta resettare la "testa" della memoria dando POKE 44,8 e dare GOTO 100.
Se in memoria, insieme alla routine, era presente un altro programma, nessuna paura, in quanto non avrà subito modifiche.

L'uso delle variabili si è reso necessario (INPUT#). Purtroppo esse andranno a modificare quelle aventi lo stesso nome e se nel programma "principale" vengono usate le variabili A, A$, B, C dopo l'attivazione della routine esse risulteranno variate, quindi attenti!

Nel caso in cui si vogliano apportare modifiche ai nomi di queste variabili, è bene sapere che la prima, terza e quarta variabile non necessariamente devono essere stringhe, in quanto i valori contenuti in esse non sono alfabetici ma numerici e non superano il valore 664. La seconda, invece, deve essere stringa in quanto contiene una serie di caratteri alfabetici (messaggio di errore). Si ricorda che la modifica della sola locazione 44 è possibile solo perché la routine proposta è particolarmente breve (=inferiore a 256 byte).

È comunque possibile allocare contemporaneamente programmi ben più lunghi a patto di calcolare accuratamente le locazioni che interessano i "puntatori" e ricordando che il byte immediatamente precedente l'inizio del Basic deve esser posto a zero. Ciò spiega, tra l'altro, la presenza di POKE 9*256,0 nella seconda riga del programma.

- Resettare il computer (spegnerlo e riaccenderlo).
- Caricare o digitare la routine pubblicata e lanciarla (RUN).
- Non appena appare il prompt (READY) digitare NEW (e premere Return) per mettere "a posto" i puntatori di sistema.
- Caricare altri programmi come di consueto.

Come funziona la procedura

Passiamo ora alla descrizione della routine che deve essere utilizzata (DOPO aver seguito i consigli precedenti) con GOTO 100 e non con RUN (pena la "perdita" delle variabili). La prima istruzione si occupa di chiudere il file N.1 eventualmente già aperto al momento dell'interruzione del programma e lo riapre assegnando il canale di errore del disco (15). In questo modo si evita un messaggio del tipo: "FILE OPEN ERROR".

Le altre tre istruzioni si occupano (OPEN) di aprire il "colloquio" col drive, di leggere (INPUT#) i messaggi di errore, di visualizzarli (PRINT) e di chiudere il canale (CLOSE). Troviamo poi una POKE e precisamente "POKE 44,9". Questa rappresenta il "segreto" della routine.

Come sappiamo, quando accendiamo il computer e lavoriamo in Basic, la "testa" della memoria utente, comincia dalla locazione 2049. Questo indirizzo è contenuto nelle locazioni 43 e 44 (43 low 44 high). Se leggiamo, mediante PEEK, tali locazioni, notiamo che nella locazione 43 è contenuto 1, mentre nella 44 è memorizzato il valore "8". Infatti:

1 + 8*256 = 2049

Quando carichiamo un programma (a meno che in "testa" non abbia particolari indirizzi di inizio) esso viene automaticamente allocato a partire dalla suddetta locazione. Se però alteriamo il contenuto delle due locazioni (43 e 44), modifichiamo la zona di memoria a disposizione del Basic facendola iniziare "dopo" la routine pubblicata.

Una volta attivata, infatti, in 43 sarà contenuto ancora il valore 1, ma in 44 il numero 9. Avremo quindi:

1 + 9*256 = 2305

che rappresenta la nuova "testa" della memoria. Se proviamo a dare un "NEW", un "LIST" o un "RUN", il computer comincerà a controllare la memoria dall'inizio, che sarà ora all'indirizzo 2305. Questo, evidentemente, non intaccherà la rou-

Programma:

```
100 CLOSE 1: OPEN 1, 8, 15: INPUT#1,A,A$,B,C
110 PRINT A;A$;B;C: CLOSE 1: POKE 44,9: POKE 9*256,0
```

Luca Cirillo

I PUNTATORI DI INIZIO E FINE BASIC

Alcuni malfunzionamenti del programma APPEND pubblicato sul N. 15 di C.C.C. offrono il pretesto per parlare ancora di quattro locazioni fondamentali per il corretto funzionamento del sistema operativo.

Premesse

Allo scopo di meglio comprendere gli argomenti trattati nel presente articolo, si rinvia il lettore ai seguenti scritti:

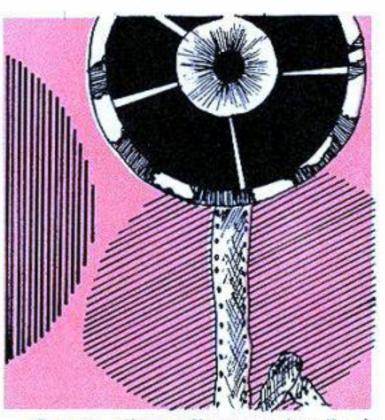
- Tecniche di overlay (N. 9)
- Un po' d'ordine tra i bit (N. 10)
- Come allungare le linee Basic (N. 10)
- Sette utility in Basic (N. 15)

Introduzione

I computer Commodore Vic 20, C-16, Commodore 64 allocano i programmi Basic, digitati da tastiera o caricati da unità magnetiche, a partire dall'indirizzo contenuto nelle due locazioni di memoria 43 e 44.

Riferendoci al Commodore 64, in tali condizioni sono normalmente contenuti i valori "1" (in 43) e "8" (in 44).
L'indirizzo (I) della prima locazione in cui verrà allocato il programma Basic sarà dato dal semplice calcolo:
I=1+8*256=2049

Per far funzionare correttamente un programma Basic è però necessario che il byte precedente tale locazione sia posto a zero. Ne potete avere facilmente conferma chiedendo, mediante PEEK, il valore ivi contenuto.

Provando, infatti, a digitare:
POKE PEEK (43) + PEEK (44) *256-1,1

Oppure, più semplicemente (per il solo C-64):
POKE 2048, 1

noterete che alcuni comandi (NEW, RUN e altri) non vengono riconosciuti e viene emessa la segnalazione SYNTAX ERROR.
Ne derivano alcune regole.

- *Prima regola.* La locazione precedente un programma Basic deve SEMPRE contenere valore nullo.
- *Seconda regola.* L'indirizzo della locazione di inizio Basic è data dal prodotto del contenuto di 43 sommata al contenuto di 44 moltiplicato per 256.

Omettendo la verifica...

- *Terza regola.* Ogni linea Basic è formata da due byte di Link (= puntatori alla prossima linea basic), due di numerazione Basic, un massimo di 80 caratteri costituenti istruzioni, comandi e dati della stessa linea basic e da un ultimo byte nullo.

L'indirizzo di questo è inferiore di un'unità al valore risultante dal calcolo dei puntatori di Link.

- *Quarta regola.* L'ultima linea di un programma basic termina, invece che con un byte nullo, con tre byte nulli, in successione.
- *Quinta regola.* L'indirizzo (I) dell'ultima locazione riservata al programma basic è anche data dal calcolo dei puntatori 45 e 46:

I=PEEK(45)+PEEK(46)*256-1

Tale indirizzo è l'ultimo byte nullo (dei tre) di cui si è parlato poc'anzi. Le didascalie di figura 1 dovrebbero eliminare ogni incertezza.

Uno strano doppione.

Da ciò che abbiamo detto sembrerebbe che il calcolatore disponga di due "sistemi" per individuare la fine di un programma:

- ☐ Indirizzo del byte puntato da 45 a 46
- ☐ Tre byte nulli successivi.

In realtà il sistema operativo del computer utilizza il primo sistema in alcuni casi ed il secondo in altri.

Quando il programma viene normalmente eseguito (e, in particolare, vengono utilizzate le istruzioni RUN, GOTO, GOSUB, eccetera), il sistema operativo provvede ad eseguire in successione i comandi contenuti nelle linee basic a partire da quella puntata da 43 a 44 e FERMANDOSI, però, nel caso in cui individua tre byte nulli in successione.

Se, invece, si registra un programma su supporto magnetico, il sistema operativo provvederà a trasferire su disco (o nastro) il contenuto di tutti i byte compresi tra i due indirizzi puntati da 43 a 44 (=inizio) e 45 e 46 (=fine).
Le didascalie della figura 2 ed i due pro-

Figura 1 zona Basic coincidente con Load e Save: A mano a mano che vengono digitate le linee basic, i due puntatori 45 e 46 vengono automaticamente aggiornati. Analogamente una routine provvede ad incorporare nella RAM la nuova linea Basic e ad inserire tre zeri, in successione, in coda ad essa (a patto che sia l'ultima, come numerazione). Il numero di linee basic, qualunque esso sia, occupa sempre due byte come pure i due byte di link. Ogni linea basic termina sempre con uno zero.

Figura 2. Manipolando opportunamente i byte 45 e 46 (=fine basic) è possibile alterare la zona RAM interessata da Load e Save che, in caso contrario, coinciderebbe con la zona contenente il solo programma basic. Nel caso illustrato in figura è possibile salvare (registrare) mediante Save, non solo il programma Basic ma anche la zona di RAM presente dopo di esso che può contenere eventualmente messaggi, variabili e, addirittura, programmi L.M. o sistemi di protezione.

grammi pubblicati, chiariscono meglio quanto esposto. L'alterazione dei quattro puntatori (43, 44, 45, 46) è la tecnica normalmente adoperata per il caricamento ed il salvataggio di programmi in Linguaggio Macchina.

Un commento a parte meritano i programmi pubblicati, scritti per un Commodore 64:
Il primo di essi, decisamente breve, occupa un numero di byte che è possibile calcolare in modo indiretto mediante FRE (0).
Le prime due righe non fanno altro che calcolare il numero di byte RAM rimasti liberi. Lo stesso risultato si ottiene digitando RUN130. Con RUN120, invece, l'alterazione "forzata" del byte 46 sottrae memoria alla RAM ed il risultato è diverso.

È ovvio che il programma sembrerà più lungo del precedente solo nel caso di registrazione (e successivo caricamento).
Nel caso normale di esecuzione di programma, invece, la sua fine verrà individuata dai tre byte nulli che sono ancora al loro posto e che non sono stati alterati minimamente dall'operazione.

Il secondo listato altera uno dei due puntatori (i lettori più pignoli possono alterare anche 45 in modo da puntare esattamente all'indirizzo desiderato). In tal modo vi sarà una zona di memoria inattaccabile dal basic in cui verrà trascritto (col semplice algoritmo di righe 110-120) un messaggio. Registrate il programma, così come è, dopo averlo fatto girare almeno una volta. Spegnete pure il computer, riaccendetelo e ricaricate il programma.
Digitate RUN 125. Vi accorgete che il messaggio viene egualmente visualizzato (a dispetto di CLR che dovrebbe cancellare ogni variabile) dato che è rimasto "incorporato", al momento della registrazione, nel programma stesso.

Tale tecnica viene normalmente adoperata per salvare, insieme al programma basic, brevi listati in linguaggio macchina oppure il nome degli autori dei programmi (in modo da renderli "indelebili") o tecniche di protezione.

Malfunzionamenti di APPEND

Da quanto esposto dovrebbe risultare chiaro che la tecnica di append, descritta nel N. 15, non fa altro che seguire la procedura descritta nelle didascalie di figura 3. Come mai, dunque, in alcuni casi il programma non funziona?
È presto detto: sono molto diffusi, tra gli appassionati di computer, particolari utility (renumber, delete ed altre) il cui funzionamento non soddisfa pienamente tali utility, al termine del loro lavoro, non fan-

Figura 3.
Funzionamento dell'utility Append: Fase 0: Accensione del computer e caricamento del programma Master. L'aggiornamento dei puntatori di fine basic è automatico (Frecce in colore). Fase 1: L'utente, altera i puntatori di inizio basic in modo che "puntino" allo stesso indirizzo di fine basic (45-46) meno due byte (a causa dei tre zeri successivi). (Freccia tratteggiata in basso). Fase 2: Il caricamento, automatico, "appende" in coda al Master, il programma desiderato (freccia tratteggiata in alto). Fase 3: Terminato il caricamento l'utente ripristina i precedenti valori di inizio basic. L'intera procedura è valida solo nel caso in cui il programma basic sia ortodosso, simile, in altre parole, a quello di figura 1 e non di figura 2.

no più coincidere l'indirizzo del byte successivo ai tre byte nulli (di fine programma) con l'indirizzo dei puntatori 45 e 46.

In genere, con tali utility, i puntatori 45 e 46 puntano qualche byte più "in là" dell'ultimo byte nullo col risultato, disastroso, di alterare completamente il sistema di linguaggio del programma slave caricato in append.
Il programma 3 risulta utile per individuare i programmi che possono generare malfunzionamenti in fase di append.

In conclusione

● Il programma Append funziona correttamente a patto che almeno il programma Master sia "ortodosso". Questo, in altre parole, deve avere coincidenti i puntatori di fine basic (45 e 46) con l'indirizzo successivo all'ultimo dei tre byte nulli.

● Nel caso in cui la fusione non avvenga, vuol dire che il programma Master non è ortodosso. In tal caso:

● Digitare il breve listato N. 3 di queste pagine (o caricarlo da nastro o disco).

Mentre tale programma è ancora visualizzato sullo schermo, caricate il programma Master non ortodosso facendo in modo che lo schermo contenga ancora il breve listato N. 3.

Terminato il caricamento risalite col cursore e premete il tasto Return su ciascuna linea Basic del programma 3: in tal modo esso verrà incolonnato in fondo al Master.

Fate partire il programma con RUN 63994.

Sullo schermo (cancellato) apparirà, dopo un tempo proporzionale alla lunghezza del programma da "aggiustare", una riga contenente quattro istruzioni (vedi riga 63998). Posizionate il cursore sul primo rigo e battete il tasto Return: in questo modo i puntatori 45 e 46 punteranno come di dovere.

Registrate il programma così modificato dopo aver cancellato le ultime righe che costituiscono il programma N. 3 (e che non servono più).

● Il programma, così registrato, è ora utilizzabile come programma Master in fusioni di Append.

Alessandro de Simone

```
100 A= FRE(0):IF A<0 THEN A=ABS(FRE(0))
110 PRINT 2↑16-A: END
120 POKE 46,9: CLR: GOTO 100
130 POKE 46,8: CLR: GOTO 100
140 :
150 REM FIGURA 1
READY.
```

```
90 PRINT FRE(0): REM VERIFICA MEMORIA PRIMA
95 POKE 46,15: CLR: RESTORE
100 PRINT FRE(0): REM VERIFICA MEMORIA DOPO
105 GET A$: IF A$="" THEN 105: REM ATTESA
110 F$= "COMMODORE COMPUTER CLUB": REM 23 CARATTERI
115 FOR I=1 TO 23
120 POKE 15*256+I, ASC(MID$(F$,I,1)):NEXT
125 PRINT CHR$(14) : REM MAIUSCOLO - MINUSCOLO
130 CLR: RESTORE: PRINT "▮▮▮▮▮": FOR I=1 TO 23
135 POKE 1024+I,PEEK(15*256+I): NEXT
140 :
145 REM FIGURA 2
READY.
```

```
63994 A1=PEEK(43)+PEEK(44)*256:A2=PEEK(45)+PEEK(46)*256
63995 FORI=A1TOA2:IFPEEK(I)=0ANDPEEK(I+1)=0ANDPEEK(I+2)=0THEN63997
63996 NEXT
63997 I1=(I+3)/256:I2=INT(I1):I1=(I1-I2)*256
63998 PRINT"▮POKE45,"I1": POKE 46,"I2":CLR:RESTORE▮▮"
63999 REM FIGURA 3
READY.
```

PAROLE CROCIATE

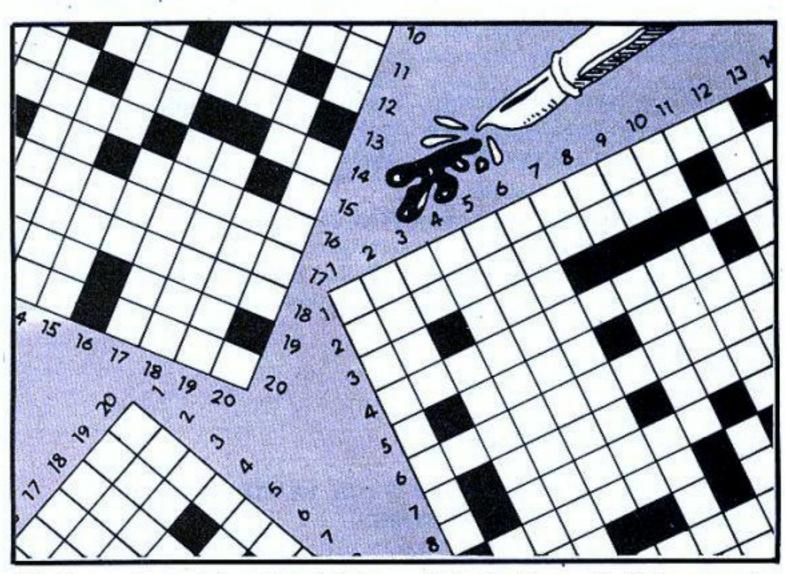
Il programma, di cui viene data la versione VIC 20 e CBM 64 è un gioco per due persone.

All'inizio viene creato uno schema di 15 per 10 caselle (caso del 64) o di 9 per 9 caselle (caso del VIC 20). Verranno poi inseriti in questo schema le "caselle nere" delle parole crociate in numero e posizione sempre casuale, così che ogni volta lo schema risulta diverso. A turno ogni giocatore inserisce una parola in orizzontale o verticale, sia da sinistra a destra, dall'alto in basso che viceversa. Cioè è possibile introdurre una parola da destra a sinistra e dal basso in alto, nella posizione voluta. Ovviamente questa parola deve essere della lunghezza giusta e, cosa molto importante, deve essere una parola esistente: non potevo introdurre un vocabolario di 20000 parole o più per controllare l'esistenza o meno di una parola!!

Il programma, quindi, si basa sulla reciproca onestà dei due giocatori. Questo fatto potrebbe, però, risultare anche positivo, per il fatto che non ci sono limitazioni e che quindi, per esempio, si può fare una sfida solo con termini inglesi o francesi ecc.

Come si usa

All'inizio viene visualizzato uno schema a quadretti, nel quale vengono introdotti, uno alla volta, i quadretti neri.

In alto a sinistra verrà infine messo il simbolo "*" della moltiplicazione. Questo sarà il nostro cursore, che muoveremo nelle quattro direzioni mediante il joystick nella porta 1 (versione C-64) o mediante i tasti funzione (versione VIC 20). Da questo punto in poi verrà fatto riferimento alla versione per il C-64.

Per il VIC valgono le stesse cose dette per il 64, tenendo presenti però le note in fondo all'articolo sulla versione VIC.

Ritorniamo al nostro cursore: appena compare significa che è iniziato il turno del primo giocatore. Dovrà posizionare il cursore nella riga e colonna in cui vuole mettere la parola, quindi deve premere il pulsante FIRE.

A questo punto, se la casella scelta è giusta, cioè non ci si è messi su una casella nera, bisogna spostare il hoystick nella direzione verso la quale intendiamo mettere la parola.

Se, per esempio, vogliamo mettere la parola in orizzontale verso sinistra, dobbiamo mettere il joystick nella direzione verso sinistra: comparirà al posto del cursore il simbolo di minore ("<") che indica la direzione scelta. Se invece vogliamo mettere la parola dal basso verso l'alto, dobbiamo spostare il joystick verso l'alto e comparirà al posto del cursore la freccia verso l'alto.

Attenzione: nella casella che precede quella scelta, nella direzione scelta, ci deve essere ovviamente una casella nera, altrimenti bisognerà posizionarsi su un'altra casella oppure cambiare direzione della parola.

Nei due esempi precedenti, allora bisogna che, nel primo, ci sia una casella nera alla destra del cursore, e nel secondo che ce ne sia una sotto di esso. Per ovvi motivi, se la casella a destra o in alto non esiste, cioè ci troviamo nel bordo dello schema, non c'è problema: verrà considerato come se ci fosse un quadrato nero. A questo punto apparirà il "ve-

ro" cursore sotto lo schema, e bisognerà introdurre la parola scelta.

Questa deve essere di lunghezza pari al numero di quadratini compresi fra il cursore ed il successivo quadratino nero nella direzione scelta, come del resto si usa in tutti i tipi di parole crociate. Inoltre se la parola incrocia altre lettere, la parola da noi introdotta deve contenere tali caratteri nella medesima posizione in cui compaiono nello schema.

Se tutto va bene la parola viene visualizzata nello schema, viene aggiornato il punteggio che appare sulla destra dello schermo, e poi viene il turno dell'altro giocatore, e così via fino alla fine.

Il punteggio viene dato in base alla lunghezza della parola introdotta. Inoltre viene dato un punteggio maggiore se si introduce una parola che ne incrocia altre. Non è permesso introdurre parole di lunghezza uno, nè è possibile riscrivere, anche la stessa parola, nella medesima posizione. Ovviamente si verranno a formare delle colonne in cui non si possono introdurre parole che abbiano senso. Questo dipende solamente dai due giocatori.

Quando tutte le caselle si riempiono o quando rimangono solo caselle da uno, o non si può più introdurre alcuna parola che abbia senso, si avvisa il computer che il gioco è terminato.

Per fare questo bisogna portarsi con il cursore su una caselle nera, quindi premere FIRE due volte di seguito. Verrà allora mostrata la classifica ed il vincitore.

Poi verrà chiesto se si vuole cominciare una nuova partita o meno, e bisogna rispondere premendo "N" o "S".

Versione VIC 20

Le differenze della versione VIC rispetto alla versione 64 sono ovviamente le dimensioni dello schema (15 per 10 nel 64 e 9 per 9 nel VIC).

Inoltre, mentre il movimento del cursore con il 64 si ottiene con il joystick, per il VIC bisogna usare i tasti funzione nel modo seguente:

F1 = alto
F3 = destra
F5 = sinistra
F7 = basso.

La barra spaziatrice, invece, è l'equivalente del tasto FIRE.

Giovanni Bellù

VERSIONE VIC 20 8/16K

```
10 REM
20 REM     PAROLE CROCIATE PER VIC+8/16K
30 REM
40 REM     GIOVANNI BELLU'  SOFTWARE 1984
50 REM
70 REM
90 REM
100 REM
110 REM SFONDO NERO
120 POKE36879,8
130 REM FUNZIONE POSIZIONE CURSORE
140 DEFFNA(A)=A+2*X+44*Y
150 REM P$ SERVE PER L'INPUT DELLE PAROLE
160 P$="■":FORK=1TO21:P$=P$+"■":NEXT
170 REM PL$CLOSESERVE PER LA VISUALIZZAZIONE DEI GIOCATORI
180 PL$="■":FORK=1TO8:PL$=PL$+"■":NEXT
190 FORK=1TO19:PL$=PL$+"■":NEXT
200 REM P1$ SERVE PER CANCELLARE LE PAROLE IN INPUT
210 FORK=1TO21:P1$=P1$+"  ":NEXT
220 REM NELLA LINEA 50 LASCIARE UNO SPAZIO FRA GLI APICI
230 REM FORMAZIONE GRIGLIA PER PAROLE CROCIATE
240 PRINT"■┌─┬─┬─┬─┬─┬─┬─┬─┬─┐"
250 FORK=1TO8
260 PRINT"│ │ │ │ │ │ │ │ │ │"
270 PRINT"├─┼─┼─┼─┼─┼─┼─┼─┼─┤"
280 NEXT
```

```
290 PRINT"| | | | | | | | | |"
300 PRINT"└┴┴┴┴┴┴┴┴┘"
310 PRINTP$"███GIOVANNI BELLU' SOFT█"
320 REM A CONTIENE L'INIZIO DEL VIDEO
330 A=4095-22
340 REM B INDICA IL NUMERO DI CASELLE NERE
350 REM B VIENE SCELTO CASUALMENTE FRA 40 E 50
360 B=INT(RND(1)*25+10)
370 IFB<25THEN360
380 REM LOOP STAMPA CASELLE NERE
390 FORK=1TOB
400 REM X = COORDINATA X DELLA K.ESIMA CASELLA NERA
410 X=INT(RND(1)*9)+1
420 REM Y = COORDINATA Y DELLA K.ESIMA CASELLA NERA
430 Y=INT(RND(1)*9)+1
440 REM CONTROLLO SE LA CASELLA E' GIA' NERA
450 IFPEEK(FNA(A))=160THEN410
460 REM STAMPA CASELLA NERA
470 POKEFNA(A),160
480 NEXT
490 X=1:Y=1:PL=1:DIMPL(2):PX=32
500 IFPEEK(FNA(A))=160THENX=X+1:GOTO500
510 REM INIZIO ROUTINE PRINCIPALE
520 REM PER DUE GIOCATORI
530 PRINTPL$"G-1"
540 PRINTPL$"█"PL(1)
550 PRINTPL$"███G-2"
560 PRINTPL$"████"PL(2)
570 IFPL=2THENPRINTPL$"████G-2":GOTO590
580 PRINTPL$"█G-1"
590 PRINTP$P1$
600 PX=PEEK(FNA(A))
610 POKEFNA(A),42
620 REM LETTURA TASTIERA
630 JK=PEEK(197):POKEFNA(A),42
640 FORK=1TO50:NEXT
650 IFJK=63THENPOKEFNA(A),PX:Y=Y+1:GOTO710
660 IFJK=39THENPOKEFNA(A),PX:Y=Y-1:GOTO730
670 IFJK=55THENPOKEFNA(A),PX:X=X-1:GOTO750
680 IFJK=47THENPOKEFNA(A),PX:X=X+1:GOTO770
690 IFJK=32THEN790
700 POKEFNA(A),PX:GOTO630
710 IFY>9THENY=9
720 GOTO520
730 IFY<1THENY=1
740 GOTO520
750 IFX<1THENX=1
760 GOTO520
770 IFX>9THENX=9
780 GOTO520
790 FORK=1TO200:NEXT:IFPX=160THEN2170
800 JK=PEEK(197)
```

```
810 FORK=1TO50:NEXT
820 REM
830 IFJK=63THENPOKEFNA(A),33:GOTO900
840 IFJK=39THENPOKEFNA(A),30:GOTO1200
850 IFJK=55THENPOKEFNA(A),60:GOTO1500
860 IFJK=47THENPOKEFNA(A),62:GOTO1800
870 IFJK=32THENPOKEFNA(A),PX:FORK=1TO200:NEXT:GOTO520
880 GOTO800
890 REM
900 REM ** PAROLA VERTICALE VERSO IL BASSO **
910 REM
920 Y=Y-1
930 IFY=0THEN950
940 IFPEEK(FNA(A))<>160THENY=Y+1:POKEFNA(A),PX:GOTO520
950 Y=Y+1
960 W=0:WX=X:WY=Y
970 WY=WY+1
980 IFWY=10THEN1000
990 WP=PEEK(A+2*WX+44*WY):IFWP<>160THEN970
1000 LN=WY-Y:W1=0:IFLN=1THEN2110
1010 CR=PEEK(FNA(A))
1020 POKEFNA(A),PX:W1=0
1030 FORK=YTOWY-1:IFPEEK(A+2*WX+44*K)<32THENW1=W1+1
1040 NEXT:IFW1=LNTHEN520
1050 POKEFNA(A),CR
1060 FORK=1TO50:GETA$:NEXT
1070 PRINTP$P1$P$;:INPUT"PAROLA";W$
1080 IFLEN(W$)<>LNTHEN1070
1090 POKEFNA(A),PX
1100 W2=0:FORK=YTOWY-1:W2=W2+1:M=ASC(MID$(W$,W2,1))-64
1110 IFM>26ORM<1THEN2100
1120 WP=PEEK(A+2*X+44*K):IFWP>29THEN1140
1130 IFWP<>MTHENPRINTP$P1$:GOTO520
1140 NEXT:POKEFNA(A),PX
1150 W2=0:PT=0:FORK=YTOWY-1:W2=W2+1:M=ASC(MID$(W$,W2,1))-64
1160 IFPEEK(A+2*X+44*K)<31THENPT=PT+4
1170 PT=PT+1:POKE(A+2*X+44*K),M:NEXT
1180 GOTO2120
1190 REM
1200 REM **  PAROLA VERTICALE VERSO L'ALTO  **
1210 REM
1220 Y=Y+1
1230 IFY=10THEN1250
1240 IFPEEK(FNA(A))<>160THENY=Y-1:POKEFNA(A),PX:GOTO520
1250 Y=Y-1
1260 W=0:WX=X:WY=Y
1270 WY=WY-1
1280 IFWY=0THEN1300
1290 WP=PEEK(A+2*WX+44*WY):IFWP<>160THEN1270
1300 LN=Y-WY:W1=0:IFLN=1THEN2110
1310 CR=PEEK(FNA(A))
1320 POKEFNA(A),PX:W1=0
```

```
1330 FORK=WY+1TOY:IFPEEK(A+2*WX+44*K)<32THENW1=W1+1
1340 NEXT:IFW1=LNTHEN520
1350 POKEFNA(A),CR
1360 FORK=1TO50:GETA$:NEXT
1370 PRINTP$P1$P$;:INPUT"PAROLA";W$
1380 IFLEN(W$)<>LNTHEN1370
1390 POKEFNA(A),PX
1400 W2=0:FORK=WY+1TOY:W2=W2+1:M=ASC(MID$(W$,LN-W2+1,1))-64
1410 IFM>26ORM<1THEN2100
1420 WP=PEEK(A+2*X+44*K):IFWP>29THEN1440
1430 IFWP<>MTHENPRINTP$P1$:GOTO520
1440 NEXT:POKEFNA(A),PX
1450 W2=0:PT=0:FORK=WY+1TOY:W2=W2+1:M=ASC(MID$(W$,LN-W2+1,1))-64
1460 IFPEEK(A+2*X+44*K)<31THENPT=PT+4
1470 PT=PT+1:POKE(A+2*X+44*K),M:NEXT
1480 GOTO2120
1490 REM
1500 REM ** PAROLA VERTICALE VERSO SINISTRA **
1510 REM
1520 X=X+1
1530 IFX=10THEN1550
1540 IFPEEK(FNA(A))<>160THENX=X-1:POKEFNA(A),PX:GOTO520
1550 X=X-1
1560 W=0:WX=X:WY=Y
1570 WX=WX-1
1580 IFWX=0THEN1600
1590 WP=PEEK(A+2*WX+44*WY):IFWP<>160THEN1570
1600 LN=X-WX:W1=0:IFLN=1THEN2110
1610 CR=PEEK(FNA(A))
1620 POKEFNA(A),PX:W1=0
1630 FORK=WX+1TOX:IFPEEK(A+2*K+44*Y)<32THENW1=W1+1
1640 NEXT:IFW1=LNTHEN520
1650 POKEFNA(A),CR
1660 FORK=1TO50:GETA$:NEXT
1670 PRINTP$P1$P$;:INPUT"PAROLA";W$
1680 IFLEN(W$)<>LNTHEN1670
1690 POKEFNA(A),PX
1700 W2=0:FORK=WX+1TOX:W2=W2+1:M=ASC(MID$(W$,LN-W2+1,1))-64
1710 IFM>26ORM<1THEN2100
1720 WP=PEEK(A+2*K+44*Y):IFWP>29THEN1740
1730 IFWP<>MTHENPRINTP$P1$:GOTO520
1740 NEXT:POKEFNA(A),PX
1750 W2=0:PT=0:FORK=WX+1TOX:W2=W2+1:M=ASC(MID$(W$,LN-W2+1,1))-64
1760 IFPEEK(A+2*K+44*Y)<31THENPT=PT+4
1770 PT=PT+1:POKE(A+2*K+44*Y),M:NEXT
1780 GOTO2120
1790 REM
1800 REM **  PAROLA VERTICALE VERSO DESTRA   **
1810 REM
1820 X=X-1
1830 IFX=0THEN1850
1840 IFPEEK(FNA(A))<>160THENX=X+1:POKEFNA(A),PX:GOTO520
```

```
1850 X=X+1
1860 W=0:WX=X:WY=Y
1870 WX=WX+1
1880 IFWX=10THEN1900
1890 WP=PEEK(A+2*WX+44*WY):IFWP<>160THEN1870
1900 LN=WX-X:W1=0:IFLN=1THEN2110
1910 CR=PEEK(FNA(A))
1920 POKEFNA(A),PX:W1=0
1930 FORK=XTOWX-1:IFPEEK(A+2*K+44*Y)<32THENW1=W1+1
1940 NEXT:IFW1=LNTHEN520
1950 POKEFNA(A),CR
1960 FORK=1TO50:GETA$:NEXT
1970 PRINTP$P1$P$;:INPUT"PAROLA";W$
1980 IFLEN(W$)<>LNTHEN1970
1990 POKEFNA(A),PX
2000 W2=0:FORK=XTOWX-1:W2=W2+1:M=ASC(MID$(W$,W2,1))-64
2010 IFM>26ORM<1THEN2100
2020 WP=PEEK(A+2*K+44*Y):IFWP>29THEN2040
2030 IFWP<>MTHENPRINTP$P1$:GOTO520
2040 NEXT:POKEFNA(A),PX
2050 W2=0:PT=0:FORK=XTOWX-1:W2=W2+1:M=ASC(MID$(W$,W2,1))-64
2060 IFPEEK(A+2*K+44*Y)<31THENPT=PT+4
2070 PT=PT+1:POKE(A+2*K+44*Y),M:NEXT
2080 GOTO2120
2090 GOTO2120
2100 K=50:W2=0:PT=0:NEXT:GOTO520
2110 POKEFNA(A),PX:GOTO520
2120 PL(PL)=PL(PL)+PT:PT=0:TT=0
2130 IFPL=1THENPL=2:PRINTPL$"G-1":PRINTPL$"■■■■G-2":GOTO2150
2140 PRINTPL$"■G-1":PRINTPL$"■■■G-2":PL=1
2150 GOTO520
2160 REM
2170 REM CONTROLLO FINE GIOCO
2180 REM
2190 TT=TT+1
2200 IFTT=2THEN2240
2210 POKEFNA(A),PX
2220 GOTO520
2230 REM
2240 REM FINE GIOCO
2250 REM
2260 PRINT"■■"
2270 IFPL(1)=PL(2)THENPRINT"■ PATTA ":GOTO2370
2280 PRINT"VINCE IL GIOCATORE";
2290 IFPL(1)>PL(2)THENPRINT1:KK=1:GOTO2320
2300 PRINT2
2310 KK=2
2320 PRINT"■CON IL PUNTEGGIO DI :"
2330 ONKKGOTO2340,2360
2340 PRINT"■■"PL(1)"■ A ■"PL(2)
2350 GOTO2370
2360 PRINT"■■"PL(2)"■ A ■"PL(1)
```

```
2370 PRINT"■■■ALTRA PARTITA ?"
2380 GETA$
2390 IFA$="S"THENRUN
2400 IFA$="N"THEN2420
2410 GOTO2380
2420 PRINT"■■":FORK=1TO50:GETA$:NEXT

READY.
```

VERSIONE COMMODORE 64

```
100 REM
110 REM        PAROLE CROCIATE PER C-64
120 REM
130 REM     GIOVANNI BELLU'  SOFTWARE 1984
140 REM
160 REM
180 REM
190 REM
200 REM SFONDO NERO
210 POKE53280,0
220 POKE53281,0
230 REM FUNZIONE POSIZIONE CURSORE
240 DEFFNA(A)=A+2*X+80*Y
250 REM P$ SERVE PER L'INPUT DELLE PAROLE
260 P$="■":FORK=1TO23:P$=P$+"■":NEXT
270 REM PL$CLOSESERVE PER LA VISUALIZZAZIONE DEI GIOCATORI
280 PL$="■":FORK=1TO8:PL$=PL$+"■":NEXT
290 FORK=1TO31:PL$=PL$+"■":NEXT
300 REM P1$ SERVE PER CANCELLARE LE PAROLE IN INPUT
310 FORK=1TO39:P1$=P1$+" ":NEXT
320 REM NELLA LINEA 50 LASCIARE UNO SPAZIO FRA GLI APICI
330 REM FORMAZIONE GRIGLIA PER PAROLE CROCIATE
340 PRINT"■■┌┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┐"
350 FORK=1TO9
360 PRINT"│││││││││││││││││││"
370 PRINT"├┼┼┼┼┼┼┼┼┼┼┼┼┼┼┼┼┤"
380 NEXT
390 PRINT"│││││││││││││││││││"
400 PRINT"└┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┘"
410 PRINTP$"■■■GIOVANNI BELLU'  SOFTWARE 1984■"
420 REM A CONTIENE L'INIZIO DEL VIDEO
430 A=1023-40
440 REM B INDICA IL NUMERO DI CASELLE NERE
450 REM B VIENE SCELTO CASUALMENTE FRA 40 E 50
460 B=INT(RND(1)*40+10)
```

```
470 IFB<40THEN460
480 REM LOOP STAMPA CASELLE NERE
490 FORK=1TOB
500 REM X = COORDINATA X DELLA K.ESIMA CASELLA NERA
510 X=INT(RND(1)*15)+1
520 REM Y = COORDINATA Y DELLA K.ESIMA CASELLA NERA
530 Y=INT(RND(1)*10)+1
540 REM CONTROLLO SE LA CASELLA E' GIA' NERA
550 IFPEEK(FNA(A))=160THEN510
560 REM STAMPA CASELLA NERA
570 POKEFNA(A),160
580 NEXT
590 X=1:Y=1:PL=1:DIMPL(2):PX=32
600 IFPEEK(FNA(A))=160THENX=X+1:GOTO600
610 REM INIZIO ROUTINE PRINCIPALE
620 REM PER DUE GIOCATORI
630 PRINTPL$"GIOC.1"
640 PRINTPL$"█PUNTI"PL(1)
650 PRINTPL$"███GIOC.2"
660 PRINTPL$"████PUNTI"PL(2)
670 IFPL=2THENPRINTPL$"████GIOC.2":GOTO690
680 PRINTPL$"█GIOC.1"
690 PRINTP$P1$
700 PX=PEEK(FNA(A))
710 POKEFNA(A),42
720 REM LETTURA JOYSTICK PORTA 1
730 JK=PEEK(56321):POKEFNA(A),42
740 FORK=1TO50:NEXT
750 IFJK=253THENPOKEFNA(A),PX:Y=Y+1:GOTO810
760 IFJK=254THENPOKEFNA(A),PX:Y=Y-1:GOTO830
770 IFJK=251THENPOKEFNA(A),PX:X=X-1:GOTO850
780 IFJK=247THENPOKEFNA(A),PX:X=X+1:GOTO870
790 IFJK=239THEN890
800 POKEFNA(A),PX:GOTO730
810 IFY>10THENY=10
820 GOTO620
830 IFY<1THENY=1
840 GOTO620
850 IFX<1THENX=1
860 GOTO620
870 IFX>15THENX=15
880 GOTO620
890 FORK=1TO200:NEXT:IFPX=160THEN2270
900 JK=PEEK(56321)
910 FORK=1TO50:NEXT
920 REM
930 IFJK=253THENPOKEFNA(A),33:GOTO1000
940 IFJK=254THENPOKEFNA(A),30:GOTO1300
950 IFJK=251THENPOKEFNA(A),60:GOTO1600
960 IFJK=247THENPOKEFNA(A),62:GOTO1900
970 IFJK=239THENPOKEFNA(A),PX:FORK=1TO200:NEXT:GOTO620
980 GOTO900
```

```
990 REM
1000 REM ** PAROLA VERTICALE VERSO IL BASSO **
1010 REM
1020 Y=Y-1
1030 IFY=0THEN1050
1040 IFPEEK(FNA(A))<>160THENY=Y+1:POKEFNA(A),PX:GOTO620
1050 Y=Y+1
1060 W=0:WX=X:WY=Y
1070 WY=WY+1
1080 IFWY=11THEN1100
1090 WP=PEEK(A+2*WX+80*WY):IFWP<>160THEN1070
1100 LN=WY-Y:W1=0:IFLN=1THEN2210
1110 CR=PEEK(FNA(A))
1120 POKEFNA(A),PX:W1=0
1130 FORK=YTOWY-1:IFPEEK(A+2*WX+80*K)<32THENW1=W1+1
1140 NEXT:IFW1=LNTHEN620
1150 POKEFNA(A),CR
1160 FORK=1TO50:GETA$:NEXT
1170 PRINTP$P1$P$;:INPUT"PAROLA";W$
1180 IFLEN(W$)<>LNTHEN1170
1190 POKEFNA(A),PX
1200 W2=0:FORK=YTOWY-1:W2=W2+1:M=ASC(MID$(W$,W2,1))-64
1210 IFM>26ORM<1THEN2200
1220 WP=PEEK(A+2*X+80*K):IFWP>29THEN1240
1230 IFWP<>MTHENPRINTP$P1$:GOTO620
1240 NEXT:POKEFNA(A),PX
1250 W2=0:PT=0:FORK=YTOWY-1:W2=W2+1:M=ASC(MID$(W$,W2,1))-64
1260 IFPEEK(A+2*X+80*K)<31THENPT=PT+4
1270 PT=PT+1:POKE(A+2*X+80*K),M:NEXT
1280 GOTO2220
1290 REM
1300 REM **  PAROLA VERTICALE VERSO L'ALTO  **
1310 REM
1320 Y=Y+1
1330 IFY=11THEN1350
1340 IFPEEK(FNA(A))<>160THENY=Y-1:POKEFNA(A),PX:GOTO620
1350 Y=Y-1
1360 W=0:WX=X:WY=Y
1370 WY=WY-1
1380 IFWY=0THEN1400
1390 WP=PEEK(A+2*WX+80*WY):IFWP<>160THEN1370
1400 LN=Y-WY:W1=0:IFLN=1THEN2210
1410 CR=PEEK(FNA(A))
1420 POKEFNA(A),PX:W1=0
1430 FORK=WY+1TOY:IFPEEK(A+2*WX+80*K)<32THENW1=W1+1
1440 NEXT:IFW1=LNTHEN620
1450 POKEFNA(A),CR
1460 FORK=1TO50:GETA$:NEXT
1470 PRINTP$P1$P$;:INPUT"PAROLA";W$
1480 IFLEN(W$)<>LNTHEN1470
1490 POKEFNA(A),PX
1500 W2=0:FORK=WY+1TOY:W2=W2+1:M=ASC(MID$(W$,LN-W2+1,1))-64
```

```
1510 IFM>26ORM<1THEN2200
1520 WP=PEEK(A+2*X+80*K):IFWP>29THEN1540
1530 IFWP<>MTHENPRINTP$P1$:GOTO620
1540 NEXT:POKEFNA(A),PX
1550 W2=0:PT=0:FORK=WY+1TOY:W2=W2+1:M=ASC(MID$(W$,LN-W2+1,1))-64
1560 IFPEEK(A+2*X+80*K)<31THENPT=PT+4
1570 PT=PT+1:POKE(A+2*X+80*K),M:NEXT
1580 GOTO2220
1590 REM
1600 REM ** PAROLA VERTICALE VERSO SINISTRA **
1610 REM
1620 X=X+1
1630 IFX=16THEN1650
1640 IFPEEK(FNA(A))<>160THENX=X-1:POKEFNA(A),PX:GOTO620
1650 X=X-1
1660 W=0:WX=X:WY=Y
1670 WX=WX-1
1680 IFWX=0THEN1700
1690 WP=PEEK(A+2*WX+80*WY):IFWP<>160THEN1670
1700 LN=X-WX:W1=0:IFLN=1THEN22:0
1710 CR=PEEK(FNA(A))
1720 POKEFNA(A),PX:W1=0
1730 FORK=WX+1TOX:IFPEEK(A+2*K+80*Y)<32THENW1=W1+1
1740 NEXT:IFW1=LNTHEN620
1750 POKEFNA(A),CR
1760 FORK=1TO50:GETA$:NEXT
1770 PRINTP$P1$P$;:INPUT"PAROLA";W$
1780 IFLEN(W$)<>LNTHEN1770
1790 POKEFNA(A),PX
1800 W2=0:FORK=WX+1TOX:W2=W2+1:M=ASC(MID$(W$,LN-W2+1,1))-64
1810 IFM>26ORM<1THEN2200
1820 WP=PEEK(A+2*K+80*Y):IFWP>29THEN1840
1830 IFWP<>MTHENPRINTP$P1$:GOTO620
1840 NEXT:POKEFNA(A),PX
1850 W2=0:PT=0:FORK=WX+1TOX:W2=W2+1:M=ASC(MID$(W$,LN-W2+1,1))-64
1860 IFPEEK(A+2*K+80*Y)<31THENPT=PT+4
1870 PT=PT+1:POKE(A+2*K+80*Y),M:NEXT
1880 GOTO2220
1890 REM
1900 REM **  PAROLA VERTICALE VERSO DESTRA  **
1910 REM
1920 X=X-1
1930 IFX=0THEN1950
1940 IFPEEK(FNA(A))<>160THENX=X+1:POKEFNA(A),PX:GOTO620
1950 X=X+1
1960 W=0:WX=X:WY=Y
1970 WX=WX+1
1980 IFWX=16THEN2000
1990 WP=PEEK(A+2*WX+80*WY):IFWP<>160THEN1970
2000 LN=WX-X:W1=0:IFLN=1THEN2210
2010 CR=PEEK(FNA(A))
2020 POKEFNA(A),PX:W1=0
```

```
2030 FORK=XTOWX-1:IFPEEK(A+2*K+80*Y)<32THENW1=W1+1
2040 NEXT:IFW1=LNTHEN620
2050 POKEFNA(A),CR
2060 FORK=1TO50:GETA$:NEXT
2070 PRINTP$P1$P$;:INPUT"PAROLA";W$
2080 IFLEN(W$)<>LNTHEN2070
2090 POKEFNA(A),PX
2100 W2=0:FORK=XTOWX-1:W2=W2+1:M=ASC(MID$(W$,W2,1))-64
2110 IFM>26ORM<1THEN2200
2120 WP=PEEK(A+2*K+80*Y):IFWP>29THEN2140
2130 IFWP<>MTHENPRINTP$P1$:GOTO620
2140 NEXT:POKEFNA(A),PX
2150 W2=0:PT=0:FORK=XTOWX-1:W2=W2+1:M=ASC(MID$(W$,W2,1))-64
2160 IFPEEK(A+2*K+80*Y)<31THENPT=PT+4
2170 PT=PT+1:POKE(A+2*K+80*Y),M:NEXT
2180 GOTO2220
2190 GOTO2220
2200 K=50:W2=0:PT=0:NEXT:GOTO620
2210 POKEFNA(A),PX:GOTO620
2220 PL(PL)=PL(PL)+PT:PT=0:TT=0
2230 IFPL=1THENPL=2:PRINTPL$"GIOC.1":PRINTPL$"GIOC.2":GOTO2250
2240 PRINTPL$"GIOC.1":PRINTPL$"GIOC.2":PL=1
2250 GOTO620
2260 REM
2270 REM CONTROLLO FINE GIOCO
2280 REM
2290 TT=TT+1
2300 IFTT=2THEN2340
2310 POKEFNA(A),PX
2320 GOTO620
2330 REM
2340 REM FINE GIOCO
2350 REM
2360 PRINT""
2370 IFPL(1)=PL(2)THENPRINT" PATTA ":GOTO2470
2380 PRINT"VINCE IL GIOCATORE";
2390 IFPL(1)>PL(2)THENPRINT1:KK=1:GOTO2420
2400 PRINT2
2410 KK=2
2420 PRINT"CON IL PUNTEGGIO DI :"
2430 ONKKGOTO2440,2460
2440 PRINT""PL(1)" A "PL(2)
2450 GOTO2470
2460 PRINT""PL(2)" A "PL(1)
2470 PRINT"ALTRA PARTITA ?"
2480 GETA$
2490 IFA$="S"THENRUN
2500 IFA$="N"THEN2520
2510 GOTO2480
2520 PRINT"":FORK=1TO50:GETA$:NEXT

READY.
```

CERCASI

La redazione di Commodore Computer Club vuole potenziarsi e ricerca collaboratori part-time e insegnanti di discipline scientifiche e tecniche preferibilmente residenti nell'area di Milano

Ai collaboratori che stiamo ricercando verrà richiesto di collaborare alle varie iniziative della casa editrice con articoli, libri, raccolte di programmi e l'italianizzazione di software, di cui abbiamo i diritti d'autore, orientati alla didattica per le scuole medie e superiori.
I prescelti, pertanto dovranno possedere un sistema completo di Vic 20 oppure Commodore 64 e sapere programmare sia in basic che in linguaggio macchina. La conoscenza dell'inglese e di altri personal computer è un titolo preferenziale.

Compensi

Tutti i lavori svolti su incarico della redazione verranno sempre compensati in base ai miglior standard di mercato.

Primo contatto

Per incontrarci telefonate allo 02/8467348 chiedendo della signorina Piera

Il Kernal

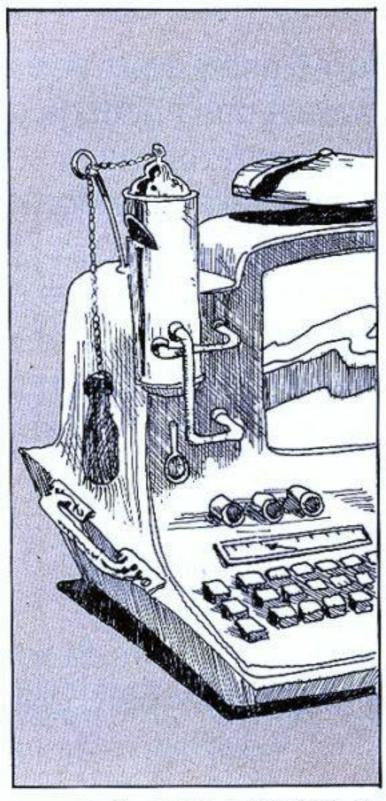
Seconda parte: lo sapevate che è possibile visualizzare più messaggi d'errore?

Sul numero scorso al quale rimando per maggiori chiarimenti, abbiamo iniziato il nostro viaggio all'interno del Kernal spiegando che esso non è altro che il sistema operativo dei computer Commodore e che le routine in linguaggio macchina che esso usa per eseguire le sue funzioni sono acessibili anche programmando in Basic o in assembly mediante dei salti in specifiche locazioni dell'ultima pagina della memoria.

Come prima sottoprocedura abbiamo esaminato la routine SCREEN, il cui compito è quello di riportare il formato dello schermo e perciò particolarmente utile per facilitare la compatibilità dei programmi.

Continuiamo la nostra analisi parlando della routine SETMSG, il cui indirizzo di chiamata si trova alla locazione 65424 ($FF90 esadecimale), avente come scopo di controllare la stampa dei messaggi del sistema per mezzo del Kernal.

A seconda del valore introdotto nell'accumulatore o, programmando in Basic, nella locazione 740 (che, come già detto la volta scorsa, ne simula la funzione) prima della chiamata della routine, si può scegliere di far stampare su video solo i messaggi d'errore, solo i messaggi di controllo, entrambi i tipi di messaggio o nessuno dei due.

A questo punto, è bene fare una precisazione: sulla Guida di Riferimento del Programmatore ed anche su altre riviste del settore, come esempio di messaggio d'errore è riportato il FILE NOT FOUND o il DIVISION BY ZERO, e come esempio di

messaggio di controllo il PRESS PLAY ON TAPE.

Ebbene, tutto questo è completamente inesatto.

Come messaggi d'errore, nella fattispecie, non si intendono quelli inviati dall'interprete quando si programma in Basic (come, ad esempio, il FILE NOT FOUND, il SYNTAX ERROR, ecc.), bensì tutti quelli che possono verificarsi usando le routine del Kernal e che si presentano, normalmente programmando in assembly, con la scritta I/O ERROR # seguita da un numero che rappresenta il codice dell'errore vero e proprio.

Ogni codice ha un proprio significato che riportiamo nella tabella di figura 1.

Così, ad esempio, se durante l'esecuzione della routine LOAD del Kernal il programma non viene trovato, su video si dovrebbe presentare, sempre operando in monitor in condizioni normali, la scritta I/O ERROR #4 che sta a segnalare proprio che il programma da noi desiderato non è presente sulla memoria di massa selezionata.

Inoltre, il PRESS PLAY ON TAPE e simili non costituiscono messaggi di controllo che possono essere controllati (scusate la ripetizione) dalla routine in questione.

In questa sede sono considerati messaggi di controllo solamente i vari SEARCHING FOR, LOADING, SAVING, ecc.), emessi durante le varie fasi del caricamento/salvataggio di file da/verso memoria di massa.

Perciò, ricapitolando, i messaggi d'errore controllati da questa routine sono solamente quelli inviati dal Kernal quando si usano delle routine di input/otuput e che, normalmente, si presentano solo in monitor di linguaggio macchina, mentre i messaggi di controllo sono soltanto quelli emessi durante le varie fasi che caratterizzano l'I/O.

Se, prima del salto alla routine, si inserisce nell'accumulatore (o nella locazione 740) il valore 64 ($40), si attiveranno solamente i messaggi d'errore; col valore uguale a 128 ($80) saranno attivati soltanto i messaggi di controllo, mentre con 192 ($C2) si abiliteranno contemporaneamente i due tipi di messaggi e con 0 ($00) si disattiveranno entrambi.

È da notare che anche in questo caso

bisognerà apportare delle correzioni alla Guida di Riferimento, dove i valori da inserire nell'accumulatore per l'attivazione delle varie scritte sono stati scambiati.

L'applicazione più sovente di questa routine nella programmazione si ha quando si intende effettuare una proiezione basata sull'auto-run (possibile solo in linguaggio macchina).

In questo caso, infatti, basta inserire le istruzioni:

LDA#$00

JSR $FF90

Preferibilmente all'inizio del programma che si occupa del LOAD e del RUN automatico, e l'utente finale non saprà mai cosa succede esattamente quando vede accendersi la spia rossa del proprio disk drive oppure quando compare su video la richiesta della pressione di alcuni tasti sul registratore.

Un'altra possibile applicazione di detta routine, questa volta però in Basic, è quella di raddoppiare il numero di messaggi di errore che compaiono sul video quando si sbagliano alcune procedure di I/O.

Se, per esempio, a floppy disk non presente o spento, digitiamo la seguente linea:

POKE 780, 192 : SYS 65424 : LOAD "P", 8

e premiamo il tasto RETURN, vedremo comparire su video:

SEARCHING FOR T

I/O ERROR #5

?DEVICE NOT PRESENT ERROR

che dimostrano come i normali messaggi dell'Interprete Basic siano integrati dal messaggio di errore del Kernal (al codice 5, infatti, corrisponde l'errore di dispositivo non presente).

Potete completare l'esame cambiando il numero 192 con gli altri valori visti prima, tenendo però ben presente che, ad eccezione dei programmi, la POKE, la SYS ed il comando basic devono essere posti sempre sulla stessa riga.

Terminata la descrizione della sottoprocedura SETMSG, trattiamo ora una routine che sortisce effetti diversi a seconda che sia impostato o meno il bit di carry.

Il termine 'bit di carry' ('bit di riporto' in italiano) è frequentemente usato nella programmazione in linguaggio macchina e, poiché una sua esauriente trattazione richiederebbe molto più spazio di quanto possibili, si consiglia la consultazione di qualche buon testo o articolo specifico per comprendere appieno il significato.

Ai nostri fini, basta soltanto sapere che, per impostare tale bit, bisogna dare l'istruzione SEC in assembly o POKE 783, PEEK (783) OR 1 in Basic, mentre per azzerarlo basta dare un CLC o un POKE 783, PEEK (783) AND 254, sempre a seconda che si tratti di programma assembly o programma basic.

La routine in questione è nota col nome PLOT ed ha l'indirizzo di chiamata in 65520 ($FFF0).

Quando il bit di carry è impostato, un salto a questa routine provoca la memorizzazione nel registro indice X (o nella locazione 781 se si sta programmando il Basic) e nel registro indice Y (o nella locazione 782 se si è in Basic) della posizione attuale del cursore sullo schermo.

Più precisamente, il registro X (o la locazione 781) conterrà il numero di riga (variabile, per il C64, tra 0 e 24) su cui è posizionato, in quel preciso istante, il cursore, mentre il registro Y (o la locazione 782) il numero di colonna (variabile tra 0 e 79).

Comunque, la vera utilità pratica di questa routine si ha quando il bit di riporto è azzerato.

In questa situazione, infatti, il salto ad essa provoca il posizionamento del cursore sulla riga specificata dal valore contenuto nel registro X (o nella locazione 781 se in Basic), valore variabile tra 0 e 24, e sulla colonna specificata dal valore del registro Y (o della locazione 782), variabile per maggiore semplicità, tra 0 e 39.

I valori 0-24 e 0-39, come forse avrete già capito, sono dettati dal fatto che lo schermo del C64 è formato da 25 righe per 40 colonne (per il Vic 20 sostituire tali valori, rispettivamente, con (0-21 e (0-22).

Utilizzando convenientemente tale routine nell'ambito della programmazione, si può risolvere senza alcuna difficoltà una delle tante carenze del Basic dei computer Commodore, la mancanza dell'istruzione

Codice	Significato
0	Routine terminata dalla pressione del tasto STOP
1	Sono stati aperti troppi file contemporaneamente
2	Si tenta di aprire un file già aperto
3	Si cerca di lavorare su un file non aperto
4	File non presente sulla periferica selezionata
5	La periferica richiesta non è presente
6	Si tenta di leggere da un file che è stato aperto per scrivere
7	Si tenta di scrivere su un file in lettura
8	Manca il nome del file per memorizzazioni su disco
9	Numero di dispositivo illegale

Figura 1: messaggi di errore che possono presentarsi [illegible]

AT (o VTAB e HTAB su altre macchine), che permette di posizionare il cursore (e, quindi, di cominciare a scrivere) in un qualsiasi punto dello schermo.

Giacché i risultati migliori si ottengono mediante la creazione di un'apposita subroutine a cui fare riferimento ogni qual volta se ne presenti la necessità, ne presentiamo due possibili, la prima in Basic e l'altra in linguaggio macchina.

Per il Basic, dopo aver stabilito che la variabile X conterrà sempre il numero di riga e che Y il numero di colonna, avremo:

```
1000 POKE783, PEEK(783) AND254
1010 POKE781, Y: POKE782, X
1020 SYS65520
1030 RETURN
```

mentre per il linguaggio macchina:

```
02A7 SEC
02A8 JSR $FFF0
02AB RTS
```

Esaminiamoli brevemente. Per il linguaggio basic, la linea 1000 assicura che il bit di carry sia azzerato, dopodiché si inseriscono nelle locazioni 781 e 782 (che simulano, rispettivamente, il registro X e il registro Y) i valori relativi al numero di riga e di colonna (rappresentati dalle variabili X e Y), seguiti dal salto all'indirizzo di chiamata della routine PLOT e dal RETURN finale.

Per il listato assembly, dopo l'istruzione che pone a 0 il bit di carry, abbiamo il salto all'indirizzo di chiamata, seguito dall'RTS che chiude la subroutine.

Per il loro uso, sarà necessario solamente inserire nelle variabili X e Y (o nei registri Y ed X se in linguaggio macchina) il numero di riga e di colonna e poi effettuare il salto alla subroutine.

Volendo, quindi, posizionare il cursore nel punto di coordinate (4,8) (riga=4 e colonna =8), per il Basic effettueremo:

```
10 X=4 : Y=8 : GOSUB 1000
20...
```

mentre per il linguaggio macchina:

```
02B0 LDX #$08
02B2 LDX #$04
02B4 JSR #$02A7
02B7...
```

Concludiamo questa seconda puntata sul Kernal esaminando la routine CLALL, il cui indirizzo di chiamata si trova in 65511 ($FFE7). Scopo di questa routine è quello di chiudere tutti i file aperti e di restituire ai canali di I/O il loro valore di default (cioè presente al momento dell'accensione).
Il dispositivo standard di input è la tastiera, mentre quello di output lo schermo.

Per il suo uso, consigliabile quando si sono aperti molti file oppure quando si è verificato un errore durante il caricamento/salvataggio di programmi da/verso memoria di massa, basta soltanto chiamare questa routine con una SYS in Basic o con un JSR in linguaggio macchina.

Perciò, in Basic avremo:

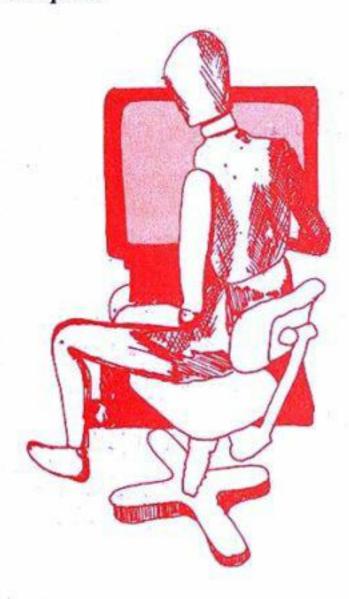
```
SYS 65511
```

mentre in linguaggio macchina:

```
JSR $FFE7
```

Sul prossimo numero, sul quale chiuderemo questa parentesi sul Kernal, tratteremo le routine LOAD, SAVE e accessorie, a mio avviso indispensabili per una avanzata conoscenza della propria macchina.

GLI ERRORI DI SINTASSI

Una breve panoramica per orientarsi correttamente nel ginepraio "Computer"

L'errore più comune che il principiante commette è sicuramente l'errore di sintassi: SYNTAX ERROR.

È necessario capire che il Basic è un linguaggio del tutto simile alla lingua degli umani e, di conseguenza, possiede regole di sintassi (di grammatica ed altre). La differenza consiste nel fatto che il computer è rigoroso, spietato e pignolo. Basta una minima differenza della sintassi corretta e il calcolatore si blocca irrimediabilmente. Il cervello umano, e la nostra educazione, consentono di comprendere i messaggi che ci rivolgono pur se contengono errori:

"Vadi a destra e, al primo semaforo, vadi sempre dritto" "Se avrei fatto una dormita, ora fossi più riposato"... ed altre, sono frasi che abbiamo spesso sentito e, nonostante tutto, perfettamente compreso. Un computer che sentisse parlare come negli esempi riportati si scandalizzerebbe immediatamente e comunicherebbe di aver incontrato difficoltà di comprensione. Purtroppo il calcolatore emette un semplice messaggio (SYNTAX ERROR IN XXX) senza indicare esattamente il posto della riga Basic in cui è esattamente ubicato l'errore né, tantomeno, il motivo.

Vedremo ora in che modo rintracciarlo all'interno di una linea. Un sistema efficace per individuare gli errori di sintassi consiste nel commetterli a bella posta ed effettuare le dovute correzioni fino a che il computer smette di visualizzare il messaggio.
Come sistema ricorreremo a numerosi esempi che meglio di un qualsiasi fiume di parole rendono chiaro il concetto di errore di sintassi:

Digitate il seguente programma:

```
100 A=123: B=A*C: C=10
110 PRINT; A PRINT; B PRINT; C
```

Approfitteremo di questo listato anche per evidenziare in seguito alcuni errori di logica. Il problema, che dovrebbe esser risolto dal programma proposto, consiste nell'eseguire un prodotto tra due valori (A e C) e visualizzarlo. C'è da ricordare che il calcolatore esegue le istruzioni l'una dopo l'altra a cominciare dalla prima incontrata.

Nel nostro caso, non appena digitiamo RUN (ed il tasto Return), il computer esamina il contenuto della riga con numerazione più bassa (100 nel nostro caso). La prima istruzione di tale riga associa alla variabile "A" il valore 123.

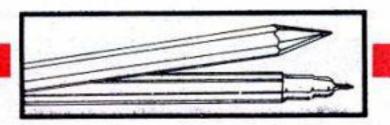
Subito dopo (notare la presenza del carattere di doppio punto [:] che separa le due istruzioni), si chiede di eseguire il prodotto di A*C e di associarlo alla variabile B. Benché l'operazione sia corretta, il programma non soddisfa il problema perché, a questo punto delle operazioni, la variabile C contiene il valore nullo (0) dato che ogni volta che si digita RUN tutte le variabili sono automaticamente poste a zero.

A nulla vale il fatto che la terza istruzione di riga 100 ponga C=10.

Tentando di fare partire il programma, otteniamo un SYNTAX ERROR IN 110 (che d'ora in poi indicheremo con S.E.). Ciò è dovuto al fatto che tra istruzioni PRINT è indispensabile inserire il carattere di doppio punto (:).

Come si può notare, il messaggio visualizzato dal computer indica anche la riga di programma Basic in cui è stato riscontrato un errore. A questo punto chiediamo il listato (LIST) e, servendoci dei tasti di controllo "saliamo" col cursore, posizioniamolo nei posti appropriati e battiamo il carattere di doppio punto ove è necessario. Non dimentichiamo di battere il tasto Return alla fine dell'operazione e, per maggior sicurezza, chiediamo nuovamente il listato per verificare che il calcolatore abbia accettato le modifiche imposte.

Digitando RUN non dovrebbero, ora, esserci inconvenienti di sorta tranne che l'errore "logico". Il vostro Commodore, infatti, stamperà, l'uno sotto l'altro, i valori 123, 0, 10 mentre il problema richiedeva come soluzione i valori 123, 1230, 10.

Per ottenere una tale risposta è necessario modificare la riga 100 come segue:

100 A=123: C=10: B=A*C

Conclusioni sulla prima esperienza.

● Tra due istruzioni è indispensabile inserire il carattere doppio punto (:)

● Il computer non può in alcun modo individuare errori logici di programmazione. L'unico modo per individuare errori di questo tipo consiste nel far "girare" il programma e nel verificare i risultati con altri già noti.

● Tra due caratteri di doppio punto (:) deve figurare un'istruzione oppure un comando completi. Esempio:

100:: PRINT A, B, C: : PRINT A*B*C: (corretto)

100: PRINT: A:B:C: PRINT: A*B:*C (ERRATO)

● Il numero di doppi punti tra due istruzioni successive non costituisce alcun.. pericolo come pure il numero di spazi tra due istruzioni o tra gli elementi di una stessa istruzione. Provate a modificare il programma precedente così:

100: : : A = 1   23: : : C = 10 : : B=A*C

L'unica differenza consiste nel fatto che il programma occupa più spazio in memoria e che il listato è di più difficile lettura. L'uso improprio del doppio punto però, in alcuni casi, facilita la lettura del listato pur se a prezzo di una maggior occupazione di memoria. La stessa riga di prima, ad esempio, si può spostare sulla destra (notare il doppio punto subito dopo la numerazione di riga):

100:          A=123:   C=10:   B=A*B

Tale comodità può esser apprezzata maggiormente nel caso in cui si disponga di una stampante.

Un altro errore logico

Rimanendo fermi sullo stesso esempio visto all'inizio, cercheremo ora di rintracciare altri errori "logici" che il computer non è in grado di individuare. In alcuni casi non è obbligatorio l'uso del punto e virgola. Modificare ad esempio la riga 110 come segue:

110 PRINT A: PRINT B: PRINT;C

Il risultato che otterrete digitando RUN (e Return) dimostra che:

● Dopo la parola-comando PRINT non è indispensabile il carattere di punto e virgola.

● Se non si inserisce quest'ultimo carattere, il computer interpreta PRINTA come se fosse presente il punto e virgola (PRINT; A).

● Non è indispensabile inserire uno spazio tra l'istruzione e la virgola (PRINTA produce lo stesso effetto di PRINT A).

Ciò premesso, sembrerebbe che il punto e virgola sia un carattere superfluo. Vedremo ora perché non è così.

Modifichiamo ora il programma come segue:

100 A=123: C=10: B=A*B
110 PRINT A;: PRINT C;: PRINT B

Grazie alla presenza dei "punto e virgola" otterremo, dopo il RUN, i tre valori affiancati anziché l'uno sotto l'altro.

Allo scopo di "risparmiare" caratteri sul listato, possiamo modificare la riga 110 come segue:

110 PRINT A; C; B

Battendo RUN e Return constatiamo, infatti, che il risultato è identico al precedente. Poiché abbiamo detto che il carattere di punto e virgola è superfluo, ci sentiamo autorizzati a digitare la riga 110 come segue:

110 PRINT A B C

Otteniamo, invece dei tre risultati di prima, un semplice zero (0) e nient'altro. Come mai?

I computer Commodore permettono di utilizzare variabili definite con uno oppure con due caratteri, come ad esempio A=10 oppure AC=15456 oppure A4=67. Variabili definite con più di due caratteri sono egualmente accettate ma memorizzate utilizzando i soli due primi caratteri.

Per esempio, digitare NU=40 oppure NUMERO=40 o ancora NUME=40 è la stessa cosa. Il calcolatore, in altre parole, assocerà alla variabile NU qualsiasi valore, calcolo o elaborazione che interesserà NU comunque venga chiamato (NU, NUM, NUMERO, NUMISANTI eccetera).

Ritornando al listato precedente, quando il computer incontra PRINT A B C

ritiene di dover visualizzare la variabile ABC che, diversa da "A", "B" oppure "C", vale zero come qualsiasi altra variabile non dichiarata esplicitamente. Ecco dunque che l'istruzione:

PRINT A; B; C

è profondamente diversa da:

PRINT A B C

proprio per la presenza del carattere di punto e virgola.

Volendo utilizzare variabili con nomi lunghi, ecco un sistema per farlo correttamente:

PRINT NUMERO; AREA; CATETO

che equivale a:

PRINT NU; AR; CA

ed è diverso, lo ripetiamo, da

PRINT NUARCA

La comprensione del listato, ovviamente, è migliore nel primo caso ed ha l'unico inconveniente di occupare più memoria.

Conclusioni sulla seconda esperienza.

- Non sempre il punto e virgola è indispensabile. Talvolta è possibile sottinderlo.
- Il punto e virgola (e, analogamente, la virgola) è indispensabile tra variabili diverse. Omettendolo possono sorgere difficoltà di interpretazione da parte del computer e portare ad effettuare elaborazioni non desiderate benché prive di errori.

Le variabili "vietate"

Abbiamo detto che è possibile utilizzare variabili indicandole con due caratteri alfabetici e, in certi casi, alfanumerici. Ce ne possiamo render conto ricorrendo ai seguenti esempi:

AA%=182: B1=43.5: CR=3*45-B1 eccetera

Non tutte le combinazioni sono però permesse. Se provate, in modo diretto oppure sotto programma, a far eseguire:

LE=18

otterrete un S.E. Il motivo è presto detto: LE sarebbe una variabile formata da due caratteri alfabetici utilizzati anche per LEN(X$), istruzione, questa, che "restituisce" la lunghezza della stringa X$ (tratteremo più avanti questo argomento).

Il computer, interpretando carattere per carattere, incontra dapprima "L", subito dopo "E" e si aspetta di incontrare una "N" (oppure una "T" dato che esiste anche l'istruzione LET).

Dato che ciò non avviene (incontra infatti il carattere "="), emette la segnalazione di errore (S.E.).

Altre variabili, i cui caratteri, d'altra parte, non coincidono con quelli iniziali di comandi o istruzioni, sono egualmente "vietate". ST è infatti riservata al Sistema Operativo (O.S.) del calcolatore per memorizzare eventuali condizioni di errore nel colloquio con il registratore. ST è, come si suol dire, una variabile a sola lettura: è possibile esaminare il contenuto ma non definirlo. In altre parole è lecito PRINT ST ma è illecito ST=46. Un'altra variabile a sola lettura è TI, utilizzata dal calcolatore per comunicare i dati relativi all'orologio interno. TI$, invece, non è vietata, è cioè alterabile a piacere dall'utente a patto che la stringa sia lunga sei caratteri.

Per esempio:

TI$="111007"

"regola" le ore 11, 10 minuti e sette secondi. TI$="11" genera, al contrario, un S.E.

Se state utilizzando particolari programmi di utility, come il Basic 4.0, altre variabili possono essere considerate vietate. Nel caso, appunto, del Basic 4.0, le variabili DS e DS$ sono variabili a sola lettura.

Come è noto, i calcolatori Commodore accettano, nella digitazione, abbreviazioni dei comandi. Per esempio invece di END si possono digitare, di seguito, i caratteri "E" ed il tasto "N" battuto insieme col tasto SHIFT. Invece di PRINT, inoltre, è sufficiente battere il punto interrogativo (?) eccetera. Siccome è possibile sbagliare nella digitazione delle abbreviazioni, il computer, incontrando un'istruzione "strana", emette il messaggio S.E. In questi casi è sufficiente visualizzare la riga indicata nel messaggio e controllarla attentamente. Esempio. Se il computer visualizza

SYNTAX ERROR IN 430

digitate LIST 430 ed esaminate la riga.

In un secondo caso l'errore non è individuabile e ciò si verifica quando si ricorre all'abbreviazione "?" nella battitura del comando PRINT# tipico delle operazioni di scrittura dati su periferiche (registazione, drive, stampante, ecc.). L'abbreviazione cui si accenna è infatti illegale: è necessario battere per esteso PRINT oppure ricorrere all'abbreviazione "P" (battuto normalmente) e tasto "R" (battuto insieme con SHIFT).

Conclusioni sulla terza esperienza

- Tutte le variabili definite con due caratteri alfabetici non possono avere le stesse iniziali dei comandi e istruzioni Basic.
- TI, ST ed altri nomi sono ugualmente vietati (variabili a sola lettura).
- TI$ deve essere costituita da sei caratteri.
- PRINT# non può essere abbreviato con ?#.

Alessandro de Simone

C'E' **commodore** AL TELEFONO

........PER FARTI PARLARE MEGLIO

DRRRR DRRRR

TU TU TU

DRHHHIN

SIGNORINA!!

VENGO

MA È MAI POSSIBILE
CHE PER UNA TELEFONATA
SI DEBBA IMPAZZIRE?!
RUBRICHE NON AGGIORNATE.....
...INDIRIZZI CAMBIATI....NUMERI
CHE NON SI TROVANO....

... INTANTO A CASA.....

TLINK!

COMO COMPUTERS

NEW FOR ITALY

...ABBIAMO IDEATO E BREVETTATO
PER VOI UN APPARECCHIO
RIVOLUZIONARIO
IL "**TELEFONO ELETTRONICO**"
.....PUÒ GESTIRE 700 UTENZE....
CHIAMARE I NUMERI....METTERLI
IN ATTESA.....
COLLEGATELO AL VOSTRO
commodore 64

'STASERA
LO DICO A PAPÀ'

commodore 64

....PENSA, È STATO
MIO FIGLIO!
HA AGGIORNATO
L'ARCHIVIO TELEFONICO
DI CASA E QUELLO
DELL'UFFICIO!
PER LUI È STATO
UN GIOCO!

......**commodore 64** È DI CASA ANCHE IN UFFICIO!

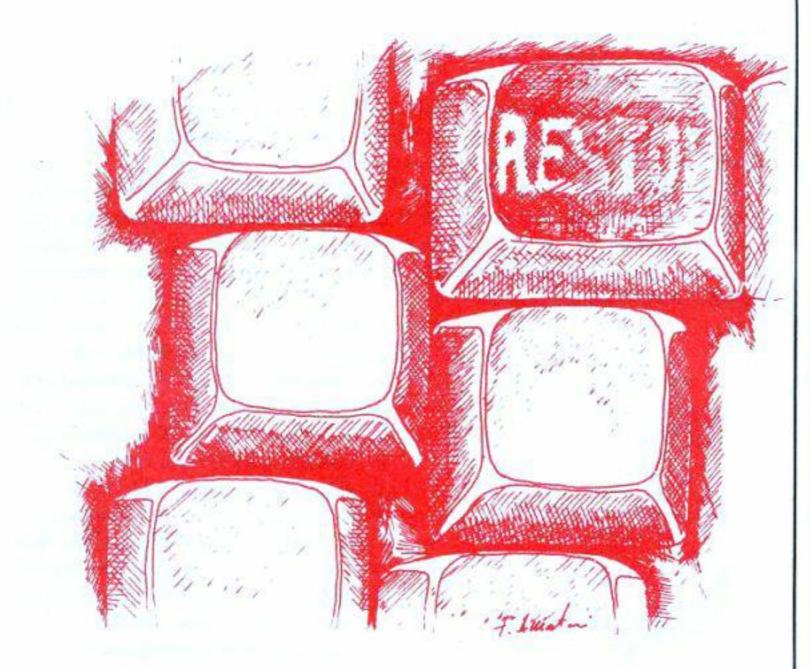
E' UN PRODOTTO HARDWARE-SOFTWARE
DELLA **COMO COMPUTERS sas**
Via Natta, 41
22100 - COMO
Tel. 031/278582 - 278876

☐ Desidero avere notizie più dettagliate sul Vs. prodotto.

Nome .... Cognome ....

Città .... CAP. ....

Via .... Tel. ....

Tasto Restore: Come Neutralizzarlo

Finalmente, dopo estenuanti attese, pubblichiamo il modo per neutralizzare i catastrofici effetti del tasto RESTORE.

Mi riferisco ovviamente al fatto che premendo tale tasto mentre è disabilitato il Sistema Operativo (quindi mentre è in esecuzione un'istruzione di tracciamento: DRAW, ARC, ecc.), si ottiene il blocco del computer, rimediabile solo con lo spegnimento. Quest'inconveniente può essere fonte di grosse arrabbiature, soprattutto per i più distratti, perciò credo che farà piacere poterlo eliminare.

Il tasto maledetto

Vediamo di capire come funziona il temibile tasto. Quando premiamo RESTORE provochiamo un NMI, cioè un interrupt non mascherabile. Ciò per il nostro microprocessore rappresenta "un ordine tassativo" (gli esperti di elettronica perdonino il mio linguaggio da autodidatta pasticcione) che non può rifiutarsi di "eseguire". Interrompe perciò qualunque cosa stesse facendo e si precipita ad eseguire la routine di NMI del Sistema Operativo, il cui indirizzo è contenuto nelle locazioni FFFA e FFFB (valori esadecimali corrispondenti ai decimali 65530 e 65531). Questa routine prevede il test del tasto STOP. Se risulta premuto, avremo come effetto alcune riinizializzazioni: delle periferiche, dei vettori principali e dello stak. Subito dopo comparirà la solita scritta READY nella parte alta dello schermo.

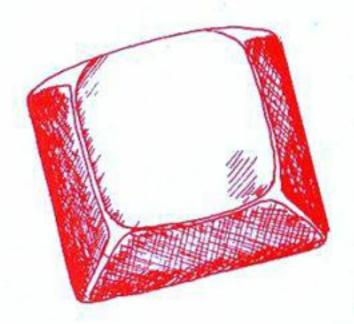
Se invece il tasto STOP non è stato premuto il microprocessore tornerà ad occuparsi di ciò che stava facendo prima dell'interruzione, con l'unica conseguenza di avere "perso" alcuni microsecondi.

Con le mie routines però (e con tante altre), durante il tracciamento di grafici in hi-res viene disabilitata la ROM che contiene il Sistema Operativo, quindi dalla locazione E000 (decimale 57344) alla FFFF (decimale 65535) il microprocessore trova memoria RAM.

Allora, quando in seguito alla pressione del tasto RESTORE il computer cerca l'indirizzo della routine di NMI... non trova ciò che cerca e va in tilt.

Disabilitare il tasto STOP (come molti hanno pensato di fare) crea altri problemi e quindi è inutile. La soluzione è molto semplice: basta creare una routine di NMI "alternativa" in una zona sicura e mettere nelle locazioni (della RAM) FFFA e FFFB l'indirizzo di tale routine. Così, anche nel caso in cui il Sistema sia momentaneamente "assente" il microprocessore saprà dove andare e cosa fare.

La nuova routine di NMI rappresenta il massimo della semplicità. È formata da una sola istruzione in linguaggio macchina: RTI che, per i profani, significa "ritorno da interrupt". Ciò significa che ogni richiesta di NMI attiva la solita procedura ma non provoca nessuna conseguenza, anche se il tasto STOP è stato premuto, perché viene subito detto al microprocessore di tornare alla sua attività ante-interrupt. L'unico effetto è la solita perdita di microsecondi per effettuare i vari salti.

Vorrei fosse chiaro che la "procedura alternativa" qui descritta viene seguita solamente quando la ROM del Sistema Operativo è disabilitata. Non si ha perciò nessuna modifica nella funzione del tasto RESTORE al di fuori del caso citato.

Le modifiche al listato

Il lavoro da compiere per una modifica di tale importanza è decisamente leggero e si riferisce, ovviamente, al programma "routine grafiche II" riportato sul n. 14 di C.C.C.

- Modificare la linea 215. L . somma di controllo C (3) passa da 17316 a 19658.
- Modificare la linea 225. La somma di controllo C (8) passa da 29161 a 29411.
- Modificare la linea 230. Il numero dopo il "TO" passa da 49873 a 49887.
- Inserire la linea ● 1155.
- Modificare la linea 1770. Gli ultimi due DATA (133 e 1) vanno sostituiti con 76 e 210.
- Modificare la linea 1780. Il primo DATA (96) va sostituito con 194.

È tutto qui. Date il RUN e poi sfogatevi premendo quante volte volete, mentre fate tracciare archi linee o altro, l'oramai innocuo RESTORE.

Danilo Toma

```
1 REM *** IL TASTO RESTORE E LE ROUTINE GRAFICHE ***
2 REM *** DI DANILO TOMA PUBLICATE SUL N.14 CCC   ***
3 :
4 REM ***  MODIFICHE E/O AGGIUNTE DA EFFETTUARE   ***
5 :
215 C=0:F1=9999:C(0)=14178:C(1)=36550:C(2)=34992:C(3)=19658:C(4)=13599
225 C(5)=27382:C(6)=4472:C(7)=29651:C(8)=29411:C(9)=34935:C(10)=18956
230 FORI=49152TO49887:GOSUB320
1155 DATA 133,1,169,223,141,250,255,169,194,141,251,255,96,64
1770 DATA 254,45,14,220,141,14,220,169,253,37,1,76,210
1780 DATA 194,169,2,5,1,133,1,169,1,13,14,220,141
READY.
```

Hard Copy della Pagina Grafica

Tutti i possessori della stampante MPS 802 hanno prima o poi desiderato, magari utilizzando il Simons' Basic o altri programmi dotati dell'opzione "hard copy", fare stampare il contenuto della pagina grafica su carta.

Purtroppo il risultato di una simile richiesta è quasi sempre un inchiodamento della stampante con la spia rossa che lampeggia minacciosamente. Le prime volte che mi è capitato questo inconveniente pensavo, come forse la maggiore parte di voi, che la stampante fosse difettosa.

Verificato poi, seguendo l'esempio riportato a pagina 23 dal manuale, che ciò non era vero, ne ho dedotto che la gestione grafica della MPS 802 è differente da quella utilizzata dalle precedenti stampanti Commodore.

L'incompatibilità può essere davvero seccante soprattutto perché "tarpa le ali" ad una macchina dalle prestazioni davvero superiori (rispetto alla MPS 801) in fatto di qualità di stampa, e che inoltre consente di utilizzare anche comuni fogli singoli. Disperarsi comunque non serve; è molto meglio copiarsi questa breve routine e inserirla nei propri programmi al posto di quelle "802-incompatibili".

La gestione della grafica della Mps 802

Come saprete, se conoscete l'inglese e se avete letto il manuale, è possibile definire e comunicare alla stampante un carattere, che non sia tra quelli del normale set, e farlo stampare.

Ogni carattere è il risultato di una matrice di 8*8 punti, e le informazioni circa i punti della matrice da scrivere o da lasciare in bianco vengono prese da otto bytes (che come è noto sono formati ognuno da otto bit). Questi devono essere inviati, tramite CHR$, alla stampante, utilizzando un apposito canale (indirizzo secondario = 5).
Il manuale spiega chiaramente come calcolare i dati da comunicare, perciò non mi dilungo.

A questo punto avrete già intuito la tecnica da utilizzare per copiare la pagina grafica (che è un insieme ordinato di bytes): basta passarne alla stampante un "pezzetto" di 8*8 punti per volta e farlo stampare come se fosse un carattere. Purtroppo sullo schermo i bytes sono posti "orizzontalmente" mentre la stampante li butta fuori "verticalmente" (vedi figura). Occorre perciò una routine che si occupi di trasformare gli otto bytes da comunicare in modo che l'uscita su carta sia corretta. È un'operazione che riesce molto più facilmente in Linguaggio Macchina (utilizzando le istruzioni di scorrimento e di rotazione) che in Basic. Sembrerebbe tutto

risolto, ma non abbiamo fatto i conti con la diabolicità dei progettisti della Commodore.

Nella nota in fondo a pagina 24 del manuale viene detto che si possono avere caratteri programmabili diversi, sulla stessa linea di stampa, solo "by overprinting". Cosa significa ciò? Che finché non mandate alla stampante il segnale di ritorno carrello CHR$(13), non potete cambiare il carattere grafico da stampare! Ma noi dobbiamo stamparne 40 di caratteri diversi per ogni riga.

La soluzione esiste, anche se è piuttosto macchinosa: basta riscrivere la stessa linea 40 volte senza fare avanzare la carta ("overprintare"), "spostando" ogni volta la posizione in cui stampare il carattere (ogni volta diverso) di una posizione verso destra.

Non spaventatevi! Il programma che svolge tale compito è più semplice della spiegazione, anche se non brilla certo per velocità di esecuzione: per stampare l'intera pagina grafica occorrono più di cinque minuti (tenete conto che vengono stampate 1000 linee). Ho parlato di sovrapposizione delle linee di stampa quindi è bene aprire una breve parentesi su questa particolare possibilità della stampante.

Mediante l'apertura di un altro degli undici canali di comunicazione (e precisamente il n. 6) si può stabilire, entro certi limiti di quanto deve avanzare la carta tra una linea e la successiva.

Un esempio di come utilizzare questa opzione si trova a pagina 25 del manuale. Passando alla stampante, tramite questo canale, un CHR$(0) otterremo di non fare avanzare la carta quando si ha il "ritorno carrello". Con CHR$(20), invece, le linee di stampa vengono poste una sotto l'altra senza lasciare spazi.

Routine di Hard Copy

Veniamo dunque al breve listato proposto. Come molti di voi sapranno è possibile mettere la pagina grafica in varie posizioni nella memoria.

La routine pubblicata stampa la pagina posizionata a partire dalla locazione 57344 (quella utilizzata dalle mie routines grafi-

che). Vi basta comunque cambiare il sesto DATA (224) della linea 10, che indica la locazione iniziale della pagina grafica, per fare stampare tutte le pagine che volete.

Il valore che dovrete inserire va calcolato così:

DATA = I/256

In cui "I" è l'indirizzo della pagina grafica.

Ad esempio il valore da sostituire a 224 sarà 160 per la pagina posizionata "sotto" l'interprete Basic (40960/256=160). Se vi può interessare il Simons' Basic utilizza la mia stessa pagina.

La routine in Linguaggio Macchina viene allocata nel buffer del registratore quindi (per sicurezza) fatela riallocare (cioè fate eseguire la linea 1000) ogni volta che utilizzate questo breve programma.

Poiché il comando RESTORE fa iniziare la lettura dal primo DATA in memoria, assicuratevi che le linee che contengono i DATA proposti siano le prime linee di DATA dei programmi in cui includete tale routine.

Un ulteriore informazione mi sembrerà doverosa: oltre alle locazioni dalla 820 alla 891 vengono utilizzate dalla routine in linguaggio macchina, per le "manipolazioni" dei dati da trasmettere, anche le locazioni dalla 892 alla 907.

Due considerazioni riguardo il formato di stampa.

La prima è che una schermata viene ridotta ad un disegno di 9.5*6 cm. (vi assicuro che la scala è 1:1). Se volete potete ottenere un disegno largo il doppio inserendo, nella linea 1060, un CHR$(14):

1060 PRINT#4, CHR$(14) B$; :
B$=B$+CHR$(32)

l'immagine risulterà però deformata (vedi figura) e occorreranno ben 23 minuti per il completamento della stampa.

Seconda considerazione. La variabile B$, introdotta nella linea 1030, determina a quale distanza dal margine sinistro del foglio deve essere stampata la pagina grafica. La distanza viene determinata dal numero di SPAZI, CHR$(32) che questa variabile contiene e ciò lo decidete voi rispondendo alla domanda che il programma vi pone.

E veniamo all'ultima linea del listato. Questa serve per resettare la stampante una volta terminata l'operazione di hard copy. È necessaria perché la stampante torni a scrivere con le linee spaziate tra loro "normalmente". In teoria basterebbe comunicare, attraverso il canale 6, il valore di "default" dell'interlinea (che a pagina 25 del manuale viene scritto essere 24) ma una prova in tale senso ha dato delle linee con una spaziatura inferiore al normale. A "occhio" il valore corretto dovrebbe essere 34 ma, a scanso d'errori, ho preferito un bel reset che aggiusta tutto.

Importante: quando vengono premuti i tasti RUN/STOP e RESTORE contemporaneamente la pagina grafica viene "sporcata" da quello che io chiamo il fantasma della scritta READY. State quindi attenti a non effettuare tale operazione prima di avere stampato il disegno.

Per dimostrare che la routine funzione, vi propongo alcuni esempi di output e i brevi programmi che li hanno generati. Il più suggestivo, cioè la funzione tridimensionale, necessita di circa un quarto d'ora per potere essere completamente plottato sul video. Tenete conto che i punti da disegnare (uno per uno) sono tanti e i calcoli complessi, anche perché i "punti nascosti" vanno saltati.

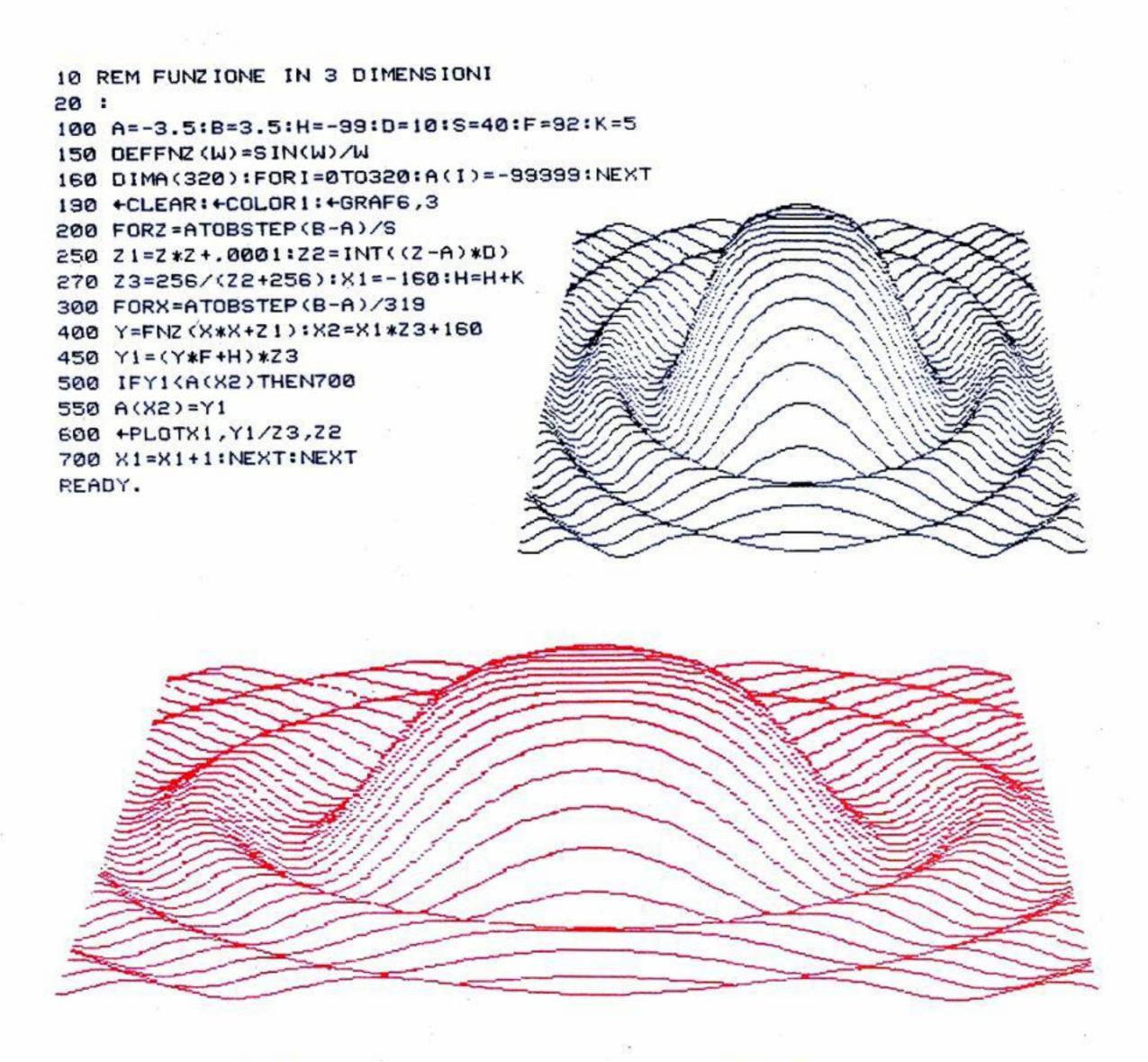
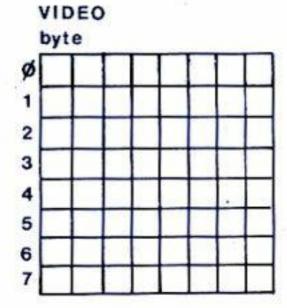
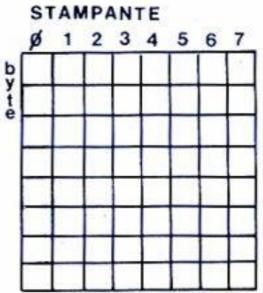
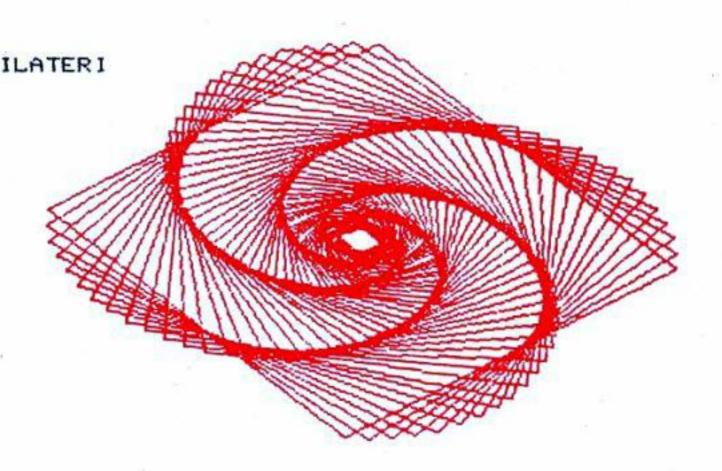
Modificando i valori dei parametri della linea 100 otterrete degli output più o meno diversi della stessa funzione.

Un'ultima (forse superflua) nota per i meno esperti. Non occorre che sia visualizzata sul monitor la pagina grafica al momento dell'esecuzione della routine di hard copy.

Riassumo infine le operazioni da effettuare per stampare la pagina grafica.

- Disegnare qualcosa nella pagina grafica.
- Caricare questo programma.
- Dare il RUN.

```
10 REM ****   CUBI   ****
20 :
30 REM SOLO PER ROUTINE GRAFICHE
50 L=120:←CLEAR:←GRAF14,0:←COLOR1
90 FORI=0TO3:READX,Y,Z
100 FORY1=YTOY-LSTEP-L
110 ←DRAWX,Y1,Z,X+L,Y1,Z
120 ←DRAW$,$,Z+L,$,$,Z+L
130 ←DRAWX+L,$,Z,$,$,$
140 ←DRAWX,$,Z,X,$,$:NEXT
170 FORZ1=ZTOZ+LSTEPL
180 ←DRAWX,Y,Z1,X,Y-L,Z1
190 ←DRAWX+L,$,$,X+L,$,$
200 NEXT:NEXT:STOP
300 DATA 39,20,0,-220,20,120
400 DATA 39,180,240,-220,180,360
READY.
```

```
10 REM FUNZIONE IN 3 DIMENSIONI
20 :
100 A=-3.5:B=3.5:H=-99:D=10:S=40:F=92:K=5
150 DEFFNZ(W)=SIN(W)/W
160 DIMA(320):FORI=0TO320:A(I)=-99999:NEXT
190 ←CLEAR:←COLOR1:←GRAF6,3
200 FORZ=ATOBSTEP(B-A)/S
250 Z1=Z*Z+.0001:Z2=INT((Z-A)*D)
270 Z3=256/(Z2+256):X1=-160:H=H+K
300 FORX=ATOBSTEP(B-A)/319
400 Y=FNZ(X*X+Z1):X2=X1*Z3+160
450 Y1=(Y*F+H)*Z3
500 IFY1<A(X2)THEN700
550 A(X2)=Y1
600 ←PLOTX1,Y1/Z3,Z2
700 X1=X1+1:NEXT:NEXT
READY.
```

```
10 REM ****  SPIRALE DI QUADRILATERI
20 ←CLEAR:←GRAF6,3:←COLOR1
30 FORI=0TO2*πSTEPπ/40
40 ←ARC0,0,I*I*80,160,
   100,I,I+2*π,2*π/4
50 NEXT:STOP
READY.
```

```
1 REM ********************************
2 REM ***                          ***
3 REM ***   (C) 1984 DANILO TOMA   ***
4 REM ***                          ***
5 REM ***     MPS 802 HARD COPY    ***
6 REM ***  DELLA PAGINA  GRAFICA   ***
7 REM ********************************
8 :
10 DATA 169,0,133,253,169,224,133,254,96,160
20 DATA 7,32,108,3,177,253,153,124,3,136
30 DATA 16,248,32,116,3,160,7,162,7,94
40 DATA 124,3,106,202,16,249,153,132,3,136
50 DATA 16,241,24,165,253,105,8,133,253,165
60 DATA 254,105,0,133,254,96,120,169,252,37
70 DATA 1,133,1,96,169,3,5,1,133,1
80 DATA 88,96
1000 RESTORE:K=0:FORI=820TO891:READA:K=K+A:POKEI,A:NEXT
1002 IFK<>8066THENPRINT"ERRORE NEI DATA":END
1005 INPUT"DISTANZA DAL MARGINE SINISTRO (0-40)";C$
1006 C=VAL(C$):IFC<0ORC>40THEN1005
1007 C$="":IFC=0THEN1010
1008 FORI=1TOC:C$=C$+CHR$(32):NEXT
1009 :
1010 OPEN4,4:OPEN5,4,5:OPEN6,4,6:SYS820
1020 FORK=0TO24:PRINT#6,CHR$(0)
1030 B$=C$:FORJ=0TO39:SYS829
1040 A$="":FORI=900TO907:A$=A$+CHR$(PEEK(I)):NEXT
1050 PRINT#5,A$
1060 PRINT#4,B$;:B$=B$+CHR$(32)
1070 PRINT#4,CHR$(254):NEXT
1080 PRINT#6,CHR$(20):PRINT#4:NEXT
1090 CLOSE4:CLOSE5:CLOSE6
1100 OPEN4,4,10:PRINT#4:CLOSE4
READY.
```

STUDIO SPRITE

Uno studio approfondito sul chip che gestisce anche il video

Una delle particolarità grafiche del commodore 64 sono gli sprite.

Penso che chiunque possieda un 64 abbia provato almeno una volta a divertirsi con gli sprite. E daremo per scontate di conseguenza, le conoscenze dei concetti più elementari ad essi legati, saprete dunque che esiste una mappa che contiene tutte le informazioni utili per la loro gestione, allocata dall'indirizzo 53248 all'indirizzo 53294 (cioè 53248+46).

Ci sono, quindi, un massimo di 47 informazioni sugli sprite. Le informazioni da 53248 a 53263 si riferiscono alle coordinate X e Y degli otto sprite disponibili. Inoltre in 53264 viene indicato se la coordinata X di un determinato sprite supera il valore di 255, cioè se è una cifra a due byte. Digitate il listato n. 1 e studiatelo attentamente. In seguito provate a visualizzare lo sprite 1 (le cui coordinate vengono indicate in 53248 per la X e 53249 per la Y) e poi provate a digitare:

POKE 53264,1

Lo sprite si sarà spostato sulla destra dello schermo di 256 Pixel dalla posizione in cui si trovava prima.

Per abilitare uno sprite bisogna modificare la locazione 53269. Per espandere lo sprite in verticale si usa la locazione 53271, mentre per espanderlo orizzontalmente bisogna modificare la locazione 53277. Il modo multicolore viene indicato nella locazione 53276.

```
10 REM *** INTRODUZIONE AGLI SPRITE ***
20 :
100 PRINT"□":REM CANCELLAZIONE SCHERMO
110 SPRITE=53248:REM INIZIO RAM PER SPRITE
115 BLOCCO=2040 :REM SELEZIONE BLOCCO N.0
120 POKE SPRITE+21,0:REM CANCELLA OGNI SPRITE
130 POKE SPRITE+23,0:REM DIMENSIONE STANDARD X
140 POKE SPRITE+29,0:REM DIMENSIONE STANDARD Y
160 POKE SPRITE+21,1:REM ABILITA SPRITE N.0
170 POKE BLOCCO,13:REM SELEZIONA BLOCCO 13.MO
180 REM SPRITE N.0 TUTTO "PIENO"
190 FOR I=0 TO 62: POKE 832+I,255: NEXT
200 POKE SPRITE +0,100:REM COORDINATA X
210 POKE SPRITE +1,150:REM COORDINATA Y
220 POKE SPRITE +39,0:REM COLORE SPRITE N.0
230 FOR J=1 TO 5000: NEXT: REM ATTESA
240 POKE SPRITE+29,1: PRINT "■ESPANSIONE ASSE X"
250 FORJ=1TO5000:NEXT:REM ATTESA
260 POKE SPRITE+23,1: PRINT"ESPANSIONE ASSE Y"
READY.
```

Ricordo che in ogni locazione si può introdurre un valore compreso fra 0 e 255 cioè un numero ad 8 bit. Ogni bit rappresenta uno sprite.

Se quindi si vuole abilitare lo sprite 1 e 2 insieme, non si deve fare:

POKE 53269,2 (viene abilitato solo lo sprite 2).

Bisogna invece fare:

POKE 53269,3

Se invece vogliamo abilitare solo lo sprite 8, dobbiamo digitare:

POKE 53269,128

(non POKE 53269,8!!!)

Questo perché per abilitare lo sprite 8, dobbiamo "accendere" l'ottavo bit, il cui valore è appunto 128.

Si riporta qui di seguito una tabella che indica il numero da introdurre nelle locazioni degli sprite.

BIT	—	VALORE
0	—	1
1	—	2
2	—	4
3	—	8
4	—	16
5	—	32
6	—	64
7	—	128

Vi sarete sicuramente accorti che il "valore" di un bit è dato da 2 elevato al numero del bit preso in considerazione (questo se cominciamo a contare i bit da 0, ovviamente). A chi è ancora alle prime armi per quanto riguarda la gestione degli sprite, consiglio di rileggere l'articolo "la gestione degli sprite" pubblicato sul numero 5 di Commodore Computer Club.

Detto questo, cominciamo a considerare due locazioni molto importanti per chi vuo-

```
100 REM  S T U D I O - S P R I T E
110 REM
120 REM          PROGRAMMA 1
280 REM  USARE I TASTI FUNZIONE:
290 REM
300 REM  F1 = IN ALTO
310 REM  F3 = A DESTRA
320 REM  F5 = A SINISTRA
330 REM  F7 = IN BASSO
340 REM
350 REM
360 PRINT"■"
370 POKE53280,0:POKE53281,0
380 PRINT"■":REM VERDE
390 REM RIEMPIMENTO SPRITE
400 FORK=832TO895
410 POKEK,255:NEXT
420 REM V=INIZIO MAPPA SPRITE
430 V=53248
440 REM SC=LOCAZIONE PER COLLISIONE SPRITE/SFONDO
450 SC=V+31
460 REM RIEMPIMENTO CASUALE VIDEO
470 FORK=1TO40
480 X=INT(RND(1)*30+1):IFX<1ORX>30THEN480
490 Y=INT(RND(1)*24+1):IFY<1ORY>24THEN490
500 POKE1023+X+40*Y,RND(1)*255+1
510 NEXT
520 REM INIT. SPRITE
530 X=100:Y=100
540 POKEV,X:POKEV+1,Y
550 POKEV+39,1:REM COLORE SPRITE 1
560 POKEV+21,1:REM ABILITA SPRITE 1
570 POKE2040,13
580 PRINT"■                    ":REM 20 SPAZI
590 REM INIZIO ROUTINE
600 PRINT"■";PEEK(SC)"   "
610 P=PEEK(197)
620 IFP=4THEN670
630 IFP=5THEN700
640 IFP=6THEN730
650 IFP=3THEN760
660 GOTO600
670 REM IN ALTO
680 Y=Y-1:IFY<30THENY=30
690 POKEV+1,Y:GOTO600
700 REM A DESTRA
710 X=X+1:IFX>255THENX=255
720 POKEV,X:GOTO600
730 REM A SINISTRA
740 X=X-1:IFX<1THENX=1
```

le usare gli sprite, soprattutto se questi vengono utilizzati in un gioco.

Queste due locazioni sono la 53278 e la 53279.

La locazione 53278 ci informa se due sprite collidono fra di loro. Al momento della collisione vengono accesi i bit corrispondenti agli sprite che sono sovrapposti. Cioè se per esempio si sovrappone lo sprite 2 e lo sprite 7, il numero presente nella locazione 53278, espresso in binario sarà: 0100010.

Il valore indicato dalla locazione 53278 sarà allora: 2↑1 + 2↑6, cioè, se vediamo la tabella di prima, 2+64 che è uguale a 66. Per leggere questo valore bisogna usare nel programma una istruzione PEEK (53278).

Attenzione, però, perché la locazione 53278 si azzera non appena viene letta, ed il valore che era in essa (nel nostro esempio 66) verrà automaticamente rimesso solo se i due sprite in questione sono ancora sovrapposti. Bisogna allora utilizzare subito l'informazione via programma (naturalmente se serve sapere al programma che gli sprite 2 e 7 si sono sovrapposti).

C'è un altro pericolo: dobbiamo azzerare (e cioè leggere) la locazione 53278 all'inizio del gioco o della routine perché se non è stata letta in precedenza, questa locazione può contenere un valore diverso da zero, cioè indica una collisione fra sprite, anche se questa è avvenuta prima dell'inizio del gioco o della routine utilizzata.

Una ulteriore limitazione sulle informazioni di collisione fra sprite è che non possiamo in alcun modo sapere da quale parte collidono i due (o più) sprite.

Infatti in qualsiasi posizione e da qualsiasi parte si tocchino gli sprite, il valore espresso nella locazione 53278 rimarrà sempre identico. Per risolvere questo problema è già pronto un programma specifico per capire dove gli sprite si toccano.

I tre programmi di cui è pubblicato il listato, invece, servono per spiegare e risolvere un problema analogo a quello spiegato prima: la collisione fra sprite e, questa volta, lo sfondo, cioè i vari caratteri che possono comparire sullo schermo.

Anche in questo caso, infatti non è pos-

sibile individuare solamente tramite la locazione 53279 (la quale indica appunto la collisione fra sprite e fondo) da quale parte lo sprite ha "toccato" un carattere. Inoltre, e questo è peggiore, non è possibile individuare con quale carattere specifico lo sprite è venuto a contatto.

In un programma, infatti, potremmo avere bisogno di far muovere lo sprite solo su determinati caratteri e non su altri. Per esempio possiamo avere la necessità di muovere lo sprite su una strada (fatta di particolari caratteri) e di capire se esso esce di strada, e cioè collide con altri caratteri (un palo della luce, ad esempio).

Sarà compito dei programmi presentati risolvere questo problema utilizzando apposite "formule magiche" mediante le quali sapere esattamente il valore di tutti i possibili caratteri (anticipo che il massimo è nove, ma questo lo vedremo dopo) che collidono (o no!!!) con un determinato sprite.

Programma 1

Questo primo programma serve per capire l'utilizzo pratico della locazione 53279 per la determinazione della collisione fra lo sprite ed il fondo. Il programma definisce uno sprite quadrato. (linee 390-410) In seguito il video viene riempito di caratteri casuali (linee 460-510)

Vengono poi definite tutte le caratteristiche dello sprite quali il colore, le coordinate X e Y (linee 520-570). Inizia quindi la routine principale del programma il quale consiste nel muovere lo sprite di un pixel alla volta usando i tasti funzione (F1 per andare in alto, F3 per andare a destra, F5 per andare a sinistra e F7 per andare in basso). Nella parte in alto a sinistra dello schermo verrà visualizzato un numero (che sarà uno oppure zero) il quale indica il valore presente nella locazione che interessa, cioè la 53279.

Vi potrete così accorgere facilmente come funzioni questa locazione: non appena lo sprite si sovrappone, almeno in un pixel, a qualche carattere, la locazione 53279 segnerà 1. Viceversa, appena ci si stacca dal carattere e lo sprite non collide

```
750 POKEV,X:GOTO600
760 REM IN BASSO
770 Y=Y+1:IFY>249THENY=249
780 POKEV+1,Y:GOTO600

READY.
```

```
100 REM  S T U D I O - S P R I T E
110 REM
120 REM
130 REM          PROGRAMMA 2
260 REM PROGRAMMA PER C= 64 + JOYSTICK
270 REM
280 REM E' UN ESEMPIO DI APPLICAZIONE
290 REM
300 REM DEL MODO DI GESTIONE DEGLI
310 REM
320 REM SPRITE SPIEGATO NELL'ARTICOLO
330 REM
340 REM
350 PRINT"[CLR][YEL]":REM CLR/HOME E GIALLO
360 REM SFONDO NERO
370 POKE53280,0
380 POKE53281,0
390 A=1026
400 REM
410 REM DEFINIZIONE FUNZIONI
420 REM PER LA GESTIONE DEGLI SPRITE
430 REM
440 DEFFNA(A)=(X-24)/8+((Y-50)/8)*40
450 DEFFNB(A)=PEEK(1024+FNA(A))
460 DEFFNC(A)=PEEK(1025+FNA(A))
470 DEFFND(A)=PEEK(1026+FNA(A))
480 DEFFNE(A)=PEEK(1064+FNA(A))
490 DEFFNF(A)=PEEK(1066+FNA(A))
500 DEFFNG(A)=PEEK(1104+FNA(A))
510 DEFFNH(A)=PEEK(1105+FNA(A))
520 DEFFNI(A)=PEEK(1106+FNA(A))
530 DEFFNL(A)=PEEK(1065+FNA(A))
540 REM
550 REM
560 X=32:Y=74:X1=X:Y1=Y
570 V=53248:FORK=VTOV+40:POKEV,0:NEXT
580 POKE2040,13:POKEV+39,1:POKEV+21,1
590 POKEV,X:POKEV+1,Y
600 POKEV+23,0
610 POKEV+29,0
620 POKEV+16,0
630 POKEV+28,0
```

```
640 REM
650 REM MEMORIZZAZIONE DATI SPRITE
660 REM
670 FORK=0TO62
680 READA:POKE832+K,A:NEXT
690 DATA255,255,255
700 DATA193,129,131
710 DATA193,129,131
720 DATA193,129,131
730 DATA193,129,131
740 DATA193,129,131
750 DATA193,129,131
760 DATA255,255,255
770 DATA193,129,131
780 DATA193,129,131
790 DATA193,129,131
800 DATA193,129,131
810 DATA193,129,131
820 DATA193,129,131
830 DATA255,255,255
840 DATA193,129,131
850 DATA193,129,131
860 DATA193,129,131
870 DATA193,129,131
880 DATA193,129,131
890 DATA255,255,255
900 REM
910 REM STAMPA PUNTI CASUALMENTE
920 REM
930 A=1024
940 FORK=1TO50
950 Y=INT(RND(1)*20+3):IFY<4ORY>22THEN950
960 X=INT(RND(1)*30+1):IFX<1ORX>30THEN950
970 IFX+40*Y>999THEN950
980 POKEA+X+40*Y,81
990 NEXT
1000 REM
1010 REM INIZIO ROUTINE PRINCIPALE
1020 REM
1030 X=32:Y=74:X1=X:Y1=Y
1040 P=PEEK(56321)
1050 IFP=254THEN1100
1060 IFP=251THEN1130
1070 IFP=247THEN1160
1080 IFP=253THEN1190
1090 GOTO1250
1100 REM IN ALTO
1110 Y=Y-8:IFY<74THENY=Y+8
1120 GOTO1210
1130 REM A SINISTRA
1140 X=X-8:IFX<32THENX=X+8
1150 GOTO1210
```

con altri caratteri, questa locazione indica zero.
Si può provare a far collidere lo sprite contemporaneamente con due o più caratteri: il risultato è sempre il medesimo: il valore sarà sempre uno, sia se lo sprite collida con un carattere, due o più.

Allo stesso modo si può osservare che il valore di questa locazione non dipende dal punto in cui è avvenuta la sovrapposizione fra lo sprite ed il fondo.

Programma 2

Questo programma è analogo al primo: c'è sempre lo sprite che si muove per il video (questa volta con l'uso del joystick, ma questo non importa), anche se non è più un quadrato, ma è un reticolato con nove quadratini (tipo quello che si usa per giocare tris, per intenderci). Il video viene ancora riempito casualmente anche se solo di pallini, il cui codice video è 81.

L'unica differenza è che lo sprite si muove di otto pixel alla volta (cioè si muove più velocemente) e nella parte in alto a sinistra del video non viene più visualizzato un solo numero, ma ve ne sono addirittura nove.

Se provate a muovervi un pò per il video con lo sprite, noterete subito che i nove numeri valgono sempre 32 oppure 81. Se avete seguito il discorso fatto fino a questo punto, avrete certamente capito che questi numeri rappresentano i caratteri che sono sotto lo sprite.

Infatti il 32, rappresenta lo spazio vuoto e l'81, come detto prima, rappresenta il pallino che abbiamo messo casualmente nel video.

Ecco spiegato perché è stato scelto uno sprite fatto di nove quadratini: in qualche modo ci si può rendere conto visivamente dei caratteri sotto lo sprite e nello stesso tempo si può verificare questo con i valori numerici visualizzati in alto a sinistra.

Provate a muovervi per lo schermo, e potrete ora capire meglio quanto detto.

Vi sarete certamente resi conto quale importanza può avere, soprattutto in un gioco, la conoscenza di queste informazioni sui caratteri presenti sotto lo sprite. In questo caso, se avete ben osservato, non

c'è bisogno che lo sprite collida con un carattere perché viene indicato in ogni caso il carattere presente sotto lo sprite (per esempio basti pensare allo spazio vuoto il cui codice video è il 32).

Spiegazione del trucco

Detto questo rimane da spiegare il modo in cui vengono analizzati i caratteri sotto lo sprite. Tutto si basa sulle coordinate X e Y dello sprite. È importante che queste siano dei multipli di 8, perché nella "formula" le coordinate vengono divise per 8, e quindi nel caso le coordinate non siano multipli di 8, il numero risultante avrà una parte decimale, mentre per istruzioni di tipo PEEK e POKE abbiamo sempre bisogno di numeri interi.

La formula principale è la seguente:

$k=(x-24)/8+((y-50)/8)*40$

o, in forma semplificata:

$k=(x-24)/8+(y-50)*5$

Nel programma questa formula è stata introdotta nell'istruzione DEF FNA (A) (linea 440), per fare in modo che l'utilizzo della formula stessa sia il più veloce possibile (non dimentichiamo che l'applicazione principale di questo metodo sono i giochi, per i quali è richiesta la massima velocità)

Analizziamo ora la formula: alla coordinata X dello sprite viene sottratto il numero 24 (multiplo di 8!), e poi il risultato viene diviso per 8. La sottrazione serve per cambiare il centro degli assi ortogonali sui quali vengono prese le coordinate X ed Y. In questo modo lo sprite non potrà avere coordinata inferiore a 32, cioé non può uscire dal video. (Questa è solo una limitazione fatta ai programmi di esempio qui riportati, ma nessuno vieta di poter cambiare il valore 24 (o 50 per la Y), a patto che sia sempre un numero multiplo di 8.

Il numero così ottenuto viene diviso per 8: notate che questo valore è proprio la grandezza in pixel di un carattere video. Potete capire questa divisione per 8 pensando che le coordinate degli sprite sono espresse in pixel, mentre le coordinate dei caratteri sono espresse in gruppo di otto pixel alla volta. Lo stesso discorso fatto per la X è valido anche per la coordinata Y dello sprite.

```
1160 REM A DESTRA
1170 X=X+8:IFX>250THENX=X-8
1180 GOTO1210
1190 REM IN BASSO
1200 Y=Y+8:IFY>220THENY=Y-8
1210 POKEV,X:POKEV+1,Y
1220 REM
1230 REM STAMPA VALORI DELLE FUNZIONI
1240 REM
1250 PRINT"■";FNB(A)FNC(A)FND(A)
1260 PRINTFNE(A)FNL(A)FNF(A)
1270 PRINTFNG(A)FNH(A)FNI(A)
1280 GOTO1040
1290 REM GIOVANNI BELLU' SOFTWARE 1984

READY.
```

```
100 REM     S T U D I O - S P R I T E
110 REM
120 REM
130 REM          TERZO PROGRAMMA
290 REM CLR HOME
300 PRINT"■■"
310 REM SFONDO NERO
320 POKE53280,0
330 POKE53281,0
340 REM RESET S.I.D.
350 FORK=54272TO52330:POKEK,0:NEXT
360 REM INIT. S.I.D.
370 POKE54296,15:REM VOLUME
380 POKE54278,240:REM A/D
390 POKE54276,33:REM FORMA D'ONDA
400 REM
410 A=1026:REM LOCAZIONE VIDEO
420 SU=54273:REM PER SUONO
430 REM
440 REM DEFINIZIONE FUNZIONI CONTROLLO
450 REM DEL CARATTERE SOTTO LO SPRITE
460 REM IN TUTTE LE DIREZIONI POSSIBILI
470 REM COMPRESE LE DIAGONALI CHE NON
480 REM VENGONO UTILIZZATE IN QUESTO
490 REM PROGRAMMA DIMOSTRATIVO.
500 REM
510 REM QUESTE LINEE SERVONO PER
520 REM QUALSIASI PROGRAMMA.
530 REM
540 DEFFNA(A)=(X-24)/8+((Y-50)/8)*40
550 DEFFNB(A)=PEEK(1024+FNA(A))
560 DEFFNC(A)=PEEK(1025+FNA(A))
```

```
570 DEFFND(A)=PEEK(1026+FNA(A))
580 DEFFNE(A)=PEEK(1064+FNA(A))
590 DEFFNF(A)=PEEK(1066+FNA(A))
600 DEFFNG(A)=PEEK(1104+FNA(A))
610 DEFFNH(A)=PEEK(1105+FNA(A))
620 DEFFNI(A)=PEEK(1106+FNA(A))
630 REM
640 REM FINE LINEE DI FUNZIONE
650 REM
660 REM
670 REM  INIT. SPRITE
680 REM
690 X=32:Y=58:X1=X:Y1=Y
700 FORK=832TO896:POKEK,255:NEXT
710 V=53248:POKEV+21,0
720 POKE2040,13:POKEV+39,2
730 POKEV,X:POKEV+1,Y
740 B=32:C=96:D=45
750 PRINT"♥GIOVANNI BELLU' : SOFTWARE 1984■■■"
760 PRINT"DIFFICOLTA' 1/2 ?"
770 GETA$:IFA$<>"1"ANDA$<>"2"THEN770
780 IFA$="1"THENC=45:GOTO790
790 POKEV+21,0
800 PRINT"♥"
810 REM
820 REM CREAZIONE CASUALE PERCORSO
830 REM
840 RN=INT(RND(1)*50+20)
850 L=4:L1=3:GOTO880
860 L=INT(RND(1)*4+1)
870 IFF=RNTHEN1270
880 ONLGOTO910,990,1070,1150
890 GOTO860
900 REM
910 REM ALTO
920 REM
930 IFL1=4THEN860
940 L1=L
950 A=A+80:FORK=1TO8:A=A-40
960 IFA<1063THENA=A+80:K=8:NEXT:GOTO860
970 GOSUB2000:POKEA,C:NEXT:F=F+1:A=A+40:GOTO860
980 REM
990 REM SINISTRA
1000 REM
1010 IFL1=3THEN860
1020 L1=L
1030 A=A+2:FORK=1TO8:A=A-1:IFINT((A-2024)/40)=(A-2024)/40THENA=A+1:GOTO1220
1040 IFA<1063THENA=A+2:K=8:NEXT:GOTO860
1050 GOSUB2000:POKEA,C:NEXT:F=F+1:A=A+1:GOTO860
1060 REM
1070 REM DESTRA
1080 REM
```

L'unica differenza è che la coordinata Y a cui è già sottratto 50, viene sì divisa per 8, ma moltiplicata per 40. Questo perché conoscendo le coordinate X e Y di un carattere sullo schermo, per poterlo "trovare" effettivamente bisogna moltiplicare la coordinata Y per 40.
Per maggiori chiarimenti, consiglio di rileggere l'articolo "un asterisco per esempio", pubblicato sul numero 15 di Commodore Computer Club.

È ovvio poi perché vengano sommati i numeri ottenuti dalla cordinata X e dalla coordinata Y dello sprite. Il numero così ottenuto servirà per poter leggere qualsiasi carattere presente sotto lo sprite.

Basta sommare infatti al numero ottenuto un altro numero per poter avere l'indirizzo di uno qualsiasi dei nove caratteri sotto lo sprite. Quest'altro numero, come si può vedere nelle linee 450-530, appartiene alla mappa video.

Per sapere il carattere presente sotto la parte in altro a sinistra dello sprite questo numero è 1024. Per sapere invece il carattere presente esattamente sotto il centro dello sprite, questo numero è 1065.

Ecco una tabella in cui vengono riportati i 9 valori necessari per trovare i corrispondenti 9 caratteri sotto lo sprite: il primo indica il numero del carattere da trovare (il numero uno indica il carattere sotto la parte alta a sinistra dello sprite, il numero nove indica il carattere presente sotto la parte destra in basso dello sprite, e così via): il secondo, invece, è il numero da sommare a quello trovato mediante la manipolazione delle coordinate dello sprite.

Carattere	—	Numero
1	—	1024
2	—	1025
3	—	1026
4	—	1064
5	—	1065
6	—	1066
7	—	1104
8	—	1105
9	—	1106

Ovviamente se vogliamo leggere uno dei caratteri presenti sotto lo sprite dobbiamo usare l'istruzione:

(valore carattere) = PEEK (numero

```
1090 IFL1=2THEN860
1100 L1=L
1110 A=A-2:FORK=1TO8:A=A+1:IFINT((A-2024-30)/40)
=(A-2024-30)/40THENA=A-1:GOTO1230
1120 IFA>1983THENA=A-2:K=8:NEXT:GOTO860
1130 GOSUB2000:POKEA,C:NEXT:F=F+1:A=A-1:GOTO860
1140 REM
1150 REM BASSO
1160 REM
1170 IFL1=1THEN860
1180 L1=L
1190 A=A-80:FORK=1TO8:A=A+40
1200 IFA>1983THENA=A-80:K=8:NEXT:GOTO860
1210 GOSUB2000:POKEA,C:NEXT:F=F+1:A=A-40:GOTO860
1220 POKEA-1,C:GOTO860
1230 POKEA+1,C:GOTO860
1240 REM
1250 REM  INIZIO GIOCO
1260 REM
1270 POKEV+21,1
1280 TI$="000000"
1290 POKEA,81:PRINT"[HOME]TEMPO ";TI$
1300 P=PEEK(56321):PRINT"[HOME]"TAB(6)TI$
1310 POKESU,0
1320 IFP=254THEN1380
1330 IFP=251THEN1470
1340 IFP=247THEN1560
1350 IFP=253THEN1650
1360 GOTO1300
1370 REM
1380 REM IN ALTO
1390 REM
1400 POKESU,50
1410 IFFNC(A)=81THEN1730:REM VITTORIA
1420 IFFNC(A)<>CTHEN1300
1430 Y=Y-8:IFFNC(A)=81THEN1730:REM VITTORIA
1440 IFFNC(A)<>CTHENY=Y+8:GOTO1300
1450 POKEV+1,Y:GOTO1300
1460 REM
1470 REM A DESTRA
1480 REM
1490 POKESU,50
1500 IFFNE(A)=81THEN1730:REM VITTORIA
1510 IFFNE(A)<>CTHEN1300
1520 X=X-8:IFFNE(A)=81THEN1730:REM VITTORIA
1530 IFFNE(A)<>CTHENX=X+8:GOTO1300
1540 POKEV,X:GOTO1300
1550 REM
1560 REM A SINISTRA
1570 REM
1580 POKESU,50
```

```
1590 IFFNF(A)=81THEN1730:REM VITTORIA
1600 IFFNF(A)<>CTHEN1300
1610 X=X+8:IFFNF(A)=81THEN1730:REM VITTORIA
1620 IFFNF(A)<>CTHENX=X-8:GOTO1300
1630 POKEV,X:GOTO1300
1640 REM
1650 REM IN BASSO
1660 REM
1670 POKESU,50
1680 IFFNH(A)=81THEN1730:REM VITTORIA
1690 IFFNH(A)<>CTHEN1300
1700 Y=Y+8:IFFNH(A)=81THEN1730:REM VITTORIA
1710 IFFNH(A)<>CTHENY=Y-8:GOTO1300
1720 POKEV+1,Y:GOTO1300
1730 REM
1740 REM VITTORIA
1750 REM
1760 POKESU,0:T$=TI$
1770 FORII=3TO8
1780 FORK=1TO255STEP2
1790 POKESU,K
1800 NEXT
1810 FORK=254TO0STEP-2
1820 POKESU,K
1830 NEXT
1840 POKEV+39,II+1
1850 NEXT
1860 POKEV+21,0
1870 REM
1880 PRINT"■TEMPO IMPIEGATO ■"T$
1890 PRINT"■■■"
1900 PRINT"ANCORA  (S/N) ?"
1910 GETA$
1920 IFA$=""THEN1910
1930 IFA$="N"THEN2030
1940 IFA$<>"S"THEN1910
1950 PRINT"■ OK -PREPARATI PER UN ALTRA PARTITA"
1960 FORK=1TO5000
1970 NEXT:REM RITARDO 5 SECONDI CIRCA
1980 RUN
1990 END
2000 IFC=45THENRETURN
2010 POKEA,D:FORII=1TO20:NEXT:RETURN
2020 REM
2030 REM FINE GIOCO
2040 REM
2050 PRINT"■ ■"
2060 POKEV+21,0
2070 END
2080 REM GIOVANNI BELLU' SOFTWARE 1984

READY.
```

calcolato).

Viceversa, se vogliamo scrivere qualche cosa sotto lo sprite (per esempio per fargli lasciare una scia), dobbiamo usare l'istruzione

POKE (numero calcolato), (valore carattere)

Se non siete sicuri di aver capito bene il funzionamento, provate ugualmente a fare delle prove, magari modificando il secondo programma o se siete più esperti, provate ad inventare un programma in cui si rende utile l'utilizzo delle formule spiegate prima (quello dello sprite che lascia una scia può essere un programma carino...)

Programma 3

Questo è un gioco vero e proprio, che utilizza il metodo spiegato di individuazione di caratteri sotto lo sprite Il gioco consiste nel condurre lo sprite (mediante l'uso del joystick) ad un pallino che verrà visualizzato sullo schermo.

Lo sprite si può muovere solo su una certa pista (cha a livello uno sarà visibile, mentre a livello due sul video comparirà solamente il pallino da raggiungere).

Non si può uscire dalla pista: questo controllo viene fatto proprio mediante la lettura dei caratteri sotto lo sprite. Nel caso del gioco "al buio", cioè nel secondo livello, il video SEMBRA tutto riempito di spazi: ci si pone la domanda di come fa il programma a capire se usciamo di pista o no. La risposta è molto semplice: il metodo impiegato è lo stesso del primo livello, solo che il carattere usato per creare il percorso non è più il segno meno (—), ma è lo spazio shiftato, che visivamente è identico allo spazio normale, ma dal punto di vista del computer, assume un codice video diverso dallo spazio normale: il percorso quindi esiste, e il computer lo vede, siamo noi che non possiamo vederlo!!

Una ulteriore applicazione di questo metodo la potrete vedere in un altro gioco, molto più divertente di questo presentato, che è da ritenersi dimostrativo.

Questo mio gioco si chiama panettone, ed è stato pubblicato sulla rivista su cassetta Commodore Club N.3.

Giovanni Bellù

SNAKE

Un serpente deve nutrirsi di cuori e picche che appaiono casualmente sullo schermo

Lo scopo è quello di mangiare più simboli possibile evitando di urtare il proprio corpo o il contorno.

Se mangiate 3 o più picche prima o durante l'apparizione dei cuori, il valore di quest'ultimi viene dimezzato. Per accedere al 2° schermo è necessario raggiungere 100.000 punti (se lo farete con più di 120.000 punti, ne avrete 10.000 in più). Attenti a non mangiare i punti di domanda: pena la morte!

Raggiunti i 200.000 punti ritornerete al primo schema ma sarete al secondo livello con punteggio e picche azzerati.

In alto a destra sono segnalati: punteggio parziale; numero picche mangiate; livello al quale vi trovate.

Andrea Tajocchi

```
100 REM *** SNEKONE PER COMMODORE 64 ***
110 REM ***     DI ANDREA TAJOCCHI     ***
150 ZV=0:Z=0:DK=DK+1
160 C=1024:CO=55296:I=127:S=32:Q=160:F=209
170 L=216:R=218:M= 20:V=41:P=1:D=1
180 U=193:O=211:PT=191:PQ=0
190 ZV=ZV+1
200 IFZV=2ANDZ>120000THENZ=Z+10000
210 CF=0 :CB=0 :CM=7 :CC=11
220 GOTO700
230 PRINT"{CLR}VELOCITA' DI GIOCO"
240 INPUT"█SCEGLI DA 1 A 9:";B:IF (B<1)+(B>9) THEN 230
250 B=306-B*34:PRINT"{CLR}"
260 REM !! CREAZIONE RETTANGOLO !!
270 POKE 53281,CF:POKE53280,CB
280 FOR W=0 TO 39:POKE CO+W,CM:POKE C+W,I
290 POKE CO+960+W,CM:POKE C+960+W,I:NEXT W
300 FOR W=1 TO 23:POKE CO+W*40,CM:POKE C+W*40,I
310 POKE CO+W*40+39,CM:POKEC+W*40+39,I:NEXT W
```

```
320 REM !! MOVIMENTO OPERATORE !!
330 GET G$:G=VAL(G$)
340 IFG$="S"THEN GOSUB1000
350 IFG$="I"THENG=2:GOTO390
360 IFG$="M"THENG=1:GOTO390
370 IFG$="J"THENG=4:GOTO390
380 IFG$="K"THENG=3
390 J=M:IF G =1 THEN GOTO 440
400 IF G=2 THEN GOTO 450
410 IF G=3 THEN GOTO 460
420 IF G=4 THEN GOTO 470
430 G=D:GOTO 390
440 M=M+40:GOTO 480
450 M=M-40:GOTO 480
460 M=M+1:GOTO 480
470 M=M-1
480 D=G:K=PEEK(C+M):IF K<>S THEN 510
490 PRINT"■■■";Z;" : ";PQ;" : ";DK:IFPQ=3THENGOSUB1320
500 POKE CO+M,1 :POKE C+M,F:FOR W=1 TO B:NEXT W:GOTO590
510 IF K=L THENZ=Z+500:GOSUB900:GOTO490
520 IFK=QANDPQ=3ORPQ>3THENZ=Z+5000:PQ=0:GOSUB900 :GOTO490
530 IF K=G THENZ=Z+10000:PQ=0:GOSUB1220:GOTO490
540 IF K=R THEN Z=Z+1000:GOSUB 900:GOTO 490
550 IF K=U THEN Z=Z+5000:PQ=PQ+1:GOSUB 900:GOTO 490
560 IF(K=Q)+(K=F)THEN680
570 IF(K=Q)+(K=PT)THEN680
580 IF (K=Q)+(K=I) THEN 680
590 FORG=0TO20:NEXTG
600 REM !! PUNTEGGIO !!
610 H=INT(RND(1)*918+41):IF PEEK(C+H)=I THEN 610
620 T=RND(1):IF T<0.035 THEN POKE C+H,R:POKE CO+H,8:GOTO 330
630 T=RND(1):IFZV=2ANDT<0.400THENPOKEC+H,PT:POKECO+H,5:GOTO330
640 T=RND(1):IF T<0.020 THEN POKE C+H,U:POKE CO+H,10:GOTO 330
650 T=RND(1)
660 IFT<0.005THEN PQ=PQ:POKEC+H,Q:POKECO+H,1:GOSUB1200:GOTO330
670 POKE CO+H,14:POKE C+H,L:GOTO 330
680 POKE CO+J,5:POKE C+J,S:POKE CO+M,2:POKE C+M,F
690 GOSUB 900:GOSUB1030:
700 REM PRESENTAZIONE
710 POKE53280,0:POKE53281,7
720 PRINTCHR$(142)"■■■          IL SERPENTONE MAGICO          ■"
730 PRINT"■GUIDA IL TUO SERPENTONE E DIVORA:        ■"
740 PRINT"■■■■ : PER AVERE  500 PUNTI"
750 PRINT"■■■■ : PER AVERE 1000 PUNTI"
760 PRINT"■■■■ : PER AVERE 5000 PUNTI"
770 PRINT"■■■■ : PER AVERE10000 PUNTI"
780 PRINT"■■NON SCONTRARTI CON IL RECINTO, CON LA "
790 PRINT"■LINEA FORMATA DAL TUO CORPO O CON I"
800 PRINT"■PUNIT DI DOMANDA"
810 PRINT"■■■   COMANDI:    ■"
820 PRINT"■■I - SU"
830 PRINT"M - GIU'"
```

```
840 PRINT"J - SINISTRA"
850 PRINT"K - DESTRA"
860 PRINT"S - PER INTERROMPERE"
870 PRINT"■■■>>>   PREMI UN TASTO PER INIZIARE   <<<■"
880 GET R$:IF R$="" THEN 880
890 GOTO230
900 REM !! SUONI !!
910 SI=54272
920 POKE SI+1,40
930 POKE SI+5,9
940 POKE SI+6,9
950 POKE SI+24,15
960 POKE SI+4,17
970 FOR DD=1 TO 20:NEXT DD
980 POKE SI+4,16
990 RETURN
1000 REM BO
1010 GETT$:IFT$=""THEN1000
1020 RETURN
1030 REM CONFRONTO
1040 IFZ>99990ANDZV=1THENGOTO160
1050 IFZV=1ANDZ<100000THENGOTO1070
1060 IFZV=2ANDZ>199900THENGOTO150
1070 IF Z>YTHENRT=0:Y=Z:GOTO1090
1080 RT=RT+1
1090 PRINT"■":FORJ=0TO39:PRINT"┌";:NEXT
1100 PRINT"■■■■SCORE : ";Z
1110 PRINT"■■■HIGH SCORE : ";Y
1120 PRINT"■■■TEMPO DET.RECORD : ";RT
1130 PRINT"■■■LIVELLO : ";DK:DK=0
1140 PRINT"■■■■■■■■■■■■■■■■■■■■■■ALTRA PARTITA(S/N)?■"
1150 PRINT"■■":FORJ=0TO39:PRINT"┌";:NEXT
1160 GET R$:IF R$="" THEN 1160
1170 IF R$="S" THEN GOTO 150
1180 IF R$="N" THEN PRINT"■ARRIVEDERCI !!!!!":END
1190 RETURN
1200 REM INGANNO
1210 FORTR=0TO1000:NEXT:ML=H:POKECO+H,0
1220 REM SUONI
1230 SI=54272
1240 POKE SI+1,110
1250 POKE SI+7,9
1260 POKE SI+9,4
1270 POKE SI+29,19
1280 POKE SI+4,17
1290 FOR DD=1 TO 20:NEXT DD
1300 POKE SI+4,16
1310 RETURN
1320 REM TOLGO PUNTI
1330 POKECO+ML,12:RETURN

READY.
```

In omaggio con l'annuario

Annuario 1985

COMPUTER

N. 73 - lire 7000 il "newsmagazine" dell'informatica

Scambiano programmi

Umberto Benelli - Viale XX Settembre 180 54031 Carrara-Avenza (MS) - Tel. 0585/57145 dalle 13 alle 23.

Francesco De Colle - P.le Capo Linaro 11/23 00053 Civitavecchia (Roma) - Tel. 0766/34171 oppure 0766/40171 ore pasti.

Gianni Gagoino - Via Graglia 18 - 10136 Torino - Tel. 011/352830 dalle 12.30 alle 14.30 oppure dalle 19.30 alle 21.30

Carlo Vincenzi - Via Resistenza 26 - 41033 Concordia s/s (MO) - Tel. 0535/54325 dalle 19 alle 22 tutti i giorni.

Paolo Anania - Via Capuana 56 - 00137 Roma - Tel. 06/823514.

Fernando Benini - Via E. Pazzi 16 - 48100 Ravenna.

Pasquale Angiuli - Via Dante 270 - 70123 Bari - Tel. 080/217806 ore 18/19.

Alvaro Ceccarini - Via Di Vittorio 10 - 58022 Follonica (GR) - Tel. 0566/43248 feriali dopo le 19.

Luca Lorenzini - Via Lumumba 11 - 41011 Campogalliano (MO) - Tel. 059/525861 dalle 14 alle 15.30 - dalle 18 alle 21.

Angelo Preatoni - Via Aurelia 198 - 17023 Ceriale (SV) - Tel.0182/90346 ore 7/22

Rollo Silano - Viale Lombardia 13 - 85029 Venosa (PZ) - Tel. 0972/32381 dopo le 15.

Umberto Crotti - Via Lidice 4 - 42015 Correggio (RE) - Tel. 0522/694144 ore 19/21.

Jacopo Mangiavacchi - Via S. A. Merici 70 00162 Roma - Tel. 8323095 ore pasti.

Orsola Lalatta - Corso Venezia 16 - 20121 Milano - Tel. 02/722368 ore 14/22 esclusi week—end.

Roberto Marciano - Corso Grosseto 170 - 10148 Torino - Tel. 011/255366 ore 18/21.

Marco Pendino - Via Ugo Betti 25 - 20100 Milano - Tel. 02/3087174.

Enzo Cati - Viale Stazione 25 - 12032 Barge (CN) - Tel. 0175/56762.

Alessandro Polo - Via Robbio 56 - 00135 Roma.

Barbara Allegretti - Via Galleria del Corso 7 05100 Terni - Tel. 0744/406155 dalle 8 alle 8.

Mario Lombardi - Via Palmanova 209 20132 Milano - Tel. 02/2567039 dalle 19.15 alle 20.15.

Patrizio Bergallo - Via Lera 29 - 10100 Torino - Tel. 011/743448 ore pasti.

Paolo Piraccini - Via Cairoli 273 - 47023 Cesena - Tel. 27651 ore pasti.

Luca Lauro - Via Vittoria Colonna 220 80077 Ischia (NA) - Tel. 081/992589 tutte le ore esclusa la mattina.

Angelo Preatoni - Via Aurelia 198 - 17023 Ceriale (SV) - Tel. 0182/90346 ore 13/23.

Claudio Pescaglia - Via N. Arata 4/8 16043 Chiavari - Tel. 0185/302491 dalle 19.30 alle 21.

Fabio Ivaldi - Via Valle Miglioretti 4 40025 Pino (TO) - Tel. 011/8440765 ore 14/19.

Gregorio Lena - Viale Silvani 3/2 - 40122 Bologna - Tel.051/551178

Norberto Binfarè - Via S.Marcellina 4 20125 Milano - Tel. 02/6432883 ore 19/21.

Roberto Quaglia - Via Martinazzoli 2 20161 Milano - Tel. 02/6462130 dalle 14-15 alle 20-22.

Roberto Consolo - Via Gherardini 6 20145 Milano - Tel. 02/389590 dalle 20.30 alle 21.30.

Sergino Leone - Viale Oleandri 13 - 80131 Napoli - Tel. 081/740133 dalle 14 alle 15.

Marcello Pettini - Via Vecchiano 10 00139 Roma - Tel 8120095 ore 20.30 in poi.

Simone Galimberti - Via Sette Ponti 56/12 52100 Arezzo (AR) - Tel.0557/381587 ore 14/21.

Vittorio Di Lorenzo - Via Italia 8 - 75015 Pisticci (MI) - Tel.0835/433041 ore 8-13.

Maurizio Sternieri - Via Gobetti 10 - 41012 Capri (MO) - Tel.059/683684 ore pasti.

Pierluigi Baldetti - Viale Glorioso 29 00153 Roma - Tel.5893406 ore 16-20.

Luca Montalbano - Via Malagrida 14 65100 Pescara - Tel. 085/34196 ore pasti.

Ciro Cane' - Via Arcora 7 - 80013 Casalnuovo (NA) - Tel. 081/8421498 dalle 17 alle 22.

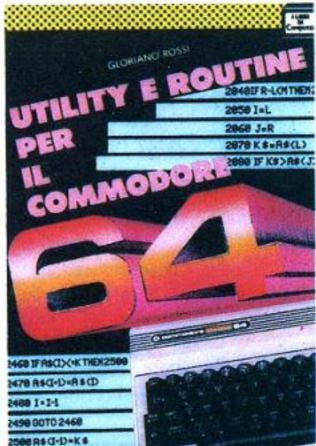
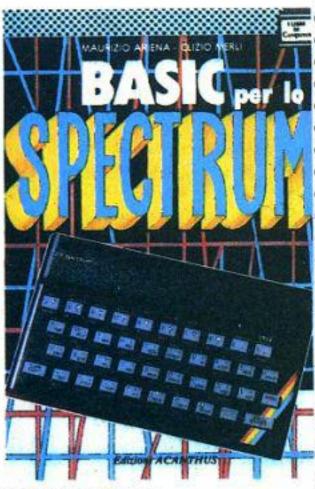
Vincenzo Petrizio - Via Nazionale 157 84030 Padula Scalo - Tel. 0975/74022 ore 13—14 e serali.

Un'iniziativa condotta con la nota rivista Computer

MICHELE DI PISA

come scegliere un COMPUTER

Guida pratica per l'acquisto di un mini o di un micro computer professionale

Edizioni ACANTHUS

PROGRAMMO IN BASIC

Il linguaggio del futuro in un manuale rapido e completo di Clizio Merli
pagg. 224 (L. 9.000)

Il Basic, attualmente il linguaggio più conosciuto - adatto all'utilizzo su qualunque tipo di macchina e in particolare sul personal e gli home-computer - può essere appreso in poche ore con l'ausilio di questo agile manuale.

COME SCEGLIERE UN COMPUTER

Guida pratica per l'acquisto di un mini o di un micro computer professionale di Michele Di Pisa
pagg. 160 (L. 6.000)

Quale modello scegliere tra gli oltre 600 computer commercializzati in Italia? La conoscenza delle caratteristiche delle varie macchine è indispensabile. Con un approccio a "menu" l'Autore vuol essere guida proprio in questa fase.

UTILITY E ROUTINE PER IL COMMODORE 64

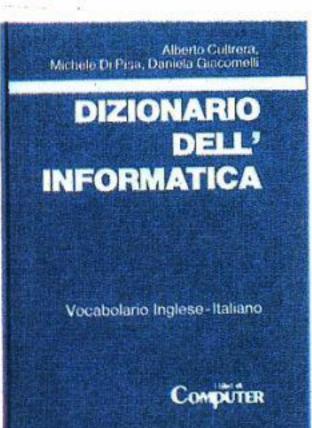
di Gloriano Rossi
pagg. 192 (L. 9.000)

L'esecuzione di una istruzione BASIC può richiedere diverse centinaia di passi di programmi in linguaggio macchina. La dimensione dei programmi è ciò che intimidisce maggiormente l'utilizzatore medio di Commodore: aiutato da questo testo chiunque potrà affrontare senza problemi il processo di scrittura di un programma.

BASIC PER LO SPECTRUM

di Maurizio Ariena e Clizio Merli
pagg. 192 (L. 9.000)

Un libro per quanti hanno acquistato il computer ZX Spectrum della Sinclair e intendono sfruttarne appieno tutte le capacità, dall'hardware alla programmazione in assembly (linguaggio macchina).

I volumi, che sono comunque in vendita nelle migliori librerie di tutta Italia, possono anche essere richiesti direttamente all'Editore.
Importante: l'ordine minimo dovrà essere di L. 15.000.

DIZIONARIO DELL'INFORMATICA

Vocabolario Inglese-Italiano di Cultrera, Di Pisa, Giacomelli
pagg. 388 (L. 25.000)

Uno strumento indispensabile per chi si avvicina al mondo dell'informatica e per gli specialisti che hanno l'esigenza di accedere alla dinamica letteratura anglosassone.

Edizioni ACANTHUS

VIALE GRAN SASSO, 23 - 20131 MILANO

Inviatemi i seguenti volumi:

Titolo	quantità	prezzo unitario
spese postali		L. 2.000
	totale	L.

Pagherò contrassegno il dovuto (più L. 2.000 per contributo spese postali) al ricevimento. Potrò restituire i libri entro 8 giorni se non saranno di mio gradimento e avere il rimborso immediato.

COGNOME ____________________

NOME ____________________

VIA ____________________ N. ______

C.A.P. __________ CITTÀ ____________________

FIRMA ____________________

DATA ____________________

Scrivere in stampatello e spedire in busta chiusa.

Sebastiano Geom. Caramagno - Via Contrada Cipollazzo / - 96011 Augusta (SR) Tel. 0931/993333.

Franco Billi - Corso Milano 36 - 14100 Asti Tel. 0141/55182 ore pasti.

Massimo Marinai - Via S. Ermete 377 56100 Pisa - Tel. 050/982656 ore pasti.

Marco Pozzesi - Via Pulci 33 - 41100 Modena - Tel. 059/337490 ore pasti.

Alessandro Grossi - Via Dario Campana 19 47037 Rimini - Tel. 0541/773458 ore 17-20.

Giuliano Robertazzi - Via Antica Posta 3 50037 S. Pietro a Sieve (FI) - Tel. 055/848393 dalle 20 in poi.

Alberto Mocellin - Via Carducci 41 - 36067 Termine di Cassola - Tel. 0424/37509 ore pasti.

Bartolomeo Aprile - Piazzetta La Maya 15 70032 Bitonto (BA) - Tel. 080/611506 ore sabato 14-18.

Leonardo De Palma - Via Francesco Marinaccio 7 - 71100 Foggia - Tel. 79700-0881 ore pomeriggio.

Luigi Monaco - Via Cutro Montedison (pal E) B - 88074 Crotone Tel. 0962/24344 ore pasti (solo nei periodi festivi).

Marco Melloni - Via Buonarotti 152 20035 Lissone (MI) - Tel.039/460853 ore pomeriggio/sera.

Paolo Gallo - Via Garibaldi 83 - 96014 Floridia (SR) - Tel. 0931/941056 dalle 14.30 alle 18.

Marcello Caputo - Via Filangieri 1 - 71100 Foggia - Tel. 0881/25072 ore pasti.

Michele Petracca - Via Donatello 12 35027 Noventa Padovana (PD) - Tel. 049/627164 ore serali.

Roberto Oselladore - Via Passo San Boldo 35/2 - 30030 Favaro Veneto (VE) - 041/631106 ore 17-18.15 - 12-13.30.

Silvano Galbiati - Via F. Meda 21 - 20050 Sovico - Tel. 039/752433 dopo le ore 19 e sabato pomeriggio.

Angelo Orlandi - Via delle Albizzie 40 00172 Roma - Tel. 06/288368 ore 16-20.

Christian Welker - Via Lagrange 11 41100 Modena - Tel. 059/357315.

Giuseppe Giorgio - Via De Giosa 48 70121 Bari - Tel. 080/544312 dopo le ore 17.30.

Stefano Castagnetti - Banzola 14 - 43013 Langhirano (PR) - Tel. 0521/853123.

Maurizio Caporale - Viale Della Rimembranza 29 - 66034 Lanciano (CH) - Tel. 0872/27296 ore pasti.

Guido Perrella - Viale C. Battisti 25 83034 Casalbore - Tel. 0825/849022 ore 16-20 feriali.

Roberto Francavilla - Via Eugenio Masi 15 71100 Foggia - Tel. 0881/75469 dalle 15 alle 17.30.

Silvano Bompieri - Via Baccaglioni 8 46040 Monzambano (MN) - Tel. 0376/845372 dopo le ore 20.

Roberto Cotza - Via Puccini 60 - 20099 Sesto San Giovanni (MI) - Tel. 02/2425392 ore 12.30-15/20.30-21.30.

Enrico Vietto - Via Torino 58 - 10090 Bruino (TO) - Tel. 011/9087197.

Franco Bellotti - Via Hermada 15 - 20162 Milano - Tel. 02/6425257.

Stefano Moltevi - Via Bragianello 6 - 22060 Figino Serenza - Tel. 031/780491 dalle 20 in poi.

Ivano Marchioro - Via Cavour 25 - 35036 Padova Tel.049/793009 dopo le ore 18.

Ivano Parbuono - Via A. di Cambio 4 37138 Verona - Tel. 045/568649 ore pasti.

Gianluca Gianluca - Via Della Scuola 51 06087 Ponte S. Giovanni (PG) - Tel. 075/395834 ore pasti o ore serali.

Andrea Bongiorni - Via Cipelli 42 - 29100 Piacenza - Tel. 0523/73071 ore pasti.

Andrea Zini - Via Predosa 11 - 40069 Zola Predosa (BO) - Tel. 051/754401 ore 20 in poi.

Michele Danese - Via G. Rossini 3 - 37044 Cologna Vi (VR) - Tel. 0442/85287 ore pomeriggio sera.

Dino Marocchi - Via Marconi 302 - 65100 Pescara - Tel. 085/68352 dopo le ore 19.

Luca Dell'Anna - Via Avellino 12 - 73100 Lecce - Tel.0832/591157 ore 14-19.

Stello Davì - Via Ogliastri 28 - 98100 Messina - Tel. 090/41822 ore pasti.

Simone Lombardi - Via V. Emanuele 205 55041 Camaiore (LV) - Tel. 0584/989728 ore pasti.

Giuseppe Borriacci - Via Mameli 15 33100 Udine - Tel. 0432/291665 ore 13-14.

Giancarlo Peruzzo - Via Harvey 2 - 35100 Padova - Tel. 049/753296 ore pasti.

Marco Belli - Via Giannone 3 - 50047 Prato (FI) - Tel. 0574/26873 ore 9.30-18.30.

Giuseppe Carollo - Via Teatro Vecchio 2 90010 Isnello (PA) - Tel. 0921/62285 ore 20.30-24.

Massimiliano Grenanin - Via XXV Aprile 4 - 10036 Settimo Torinese (TO) - Tel. 011/8010392 ore 18-20.

Andrea Burri - Via Panfili 11 - 40133 Bologna - Tel. 051/563860 ore pasti (non di mercoledì).

Gabriele Tedeschi - Via Giotto 7 - 43036 Fidenza - Tel. 0524/2213 ore pasti.

Francesco Arcidiacono - Via Acquedotto del Peschiera 36 - 00135 Roma - Tel. 06/334746.

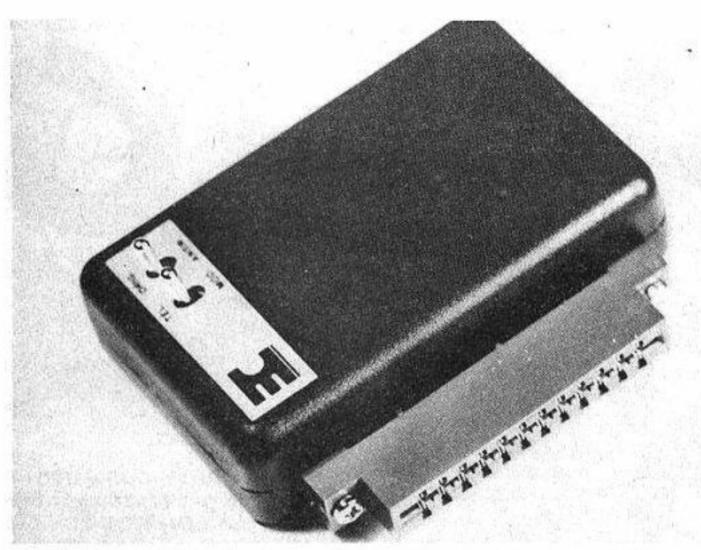
Claudio Fele — Via Delle Azzorre 391 - 00121 Ostia (Roma) - Tel. 06/5615373 ore 19—21.

Paolo Cangionelli - Via Oslavia 43 - 00195 Roma - Tel. 06/385173 ore 13-15 o 20-21.

Luciano Pagnin - Via Castello 2084 - 30122 Venezia - Tel. 041/700486 dopo le ore 20.30.

Paolo Solano - P.zza Medaglie D'Oro 13 14100 Asti - Tel. 0141/51973 ore 18-22.

Mariano Bonaccorso - Via Torrione 42 89100 Reggio Calabria - Tel. 0965/22530 ore 14.30-18.30.

Il Modem della B & C

Forse quando si parla di modem non se ne intende in verità la reale funzione e significato. La parola Modem è costruita su altre due parole: Modulatore e Demodulatore. La prima intende la parte trasmittente del sistema, mentre la seconda dà per la parte ricevente. Ma-...cosa trasmettiamo e cosa riceviamo, e come avviene il trasferimento?

Ogni computer può essere equipaggiato di un aggeggino che opportunatamente pilotato riduce in inpulsi elettrici o fonici un segnale esclusivamente digitale. Con questo tipo di segnale si possono eseguire molteplici scambi di informazioni come ad esempio: messaggi colloquiali, file di dati, programmi e comandi. Dal punto di vista hardware la B & C elettronica inserisce "lo scatolino" nero nella user-port del Commodore. Basta quindi agganciare, con due semplici fili, la linea telefonica, ed ecco che con il software fornito è già possibile comunicare con un corrispondente che abbia anch'esso un sistema Modem.

Con il Kit B & C è possibile eseguire collegamenti per scambi di messaggi e dati espressi in forma sequenziale. Ecco che sarà anche possibile iniziare programmi in BASIC, sempre se il listato relativo sia stato creato su disco piuttosto che su carta.

Come si fa? Ecco:

```
OPEN2,8,2,"0:LISTATO,S,W"
CMD2
LIST
```

Quando riapparirà il cursore si dovrà chiudere in questa esatta maniera:

```
PRINT#2:CLOSE2
```

Avremo a questo punto un file SEQuenziale che sarà l'esatta immagine del listato del programma da trasmettere. L'utility del Modem prevede l'input di un fil e SEQuenziale e quindi ogni commento ulteriore è superfluo. Come potrà avvenire un collegamento tipo con un corrispondente? La procedura pratica da seguire sarà molto semplice.

Dopo aver composto il numero telefonico occorre stabilire con il corrispondente i parametri tecnici di comunicazione. Il Baudrate (300 di norma), la lunghezza della parola (7 o 8 bit) etc. e... prima di iniziare il vero e proprio collegamento si stabilirà chi dei due sarà l'originate e chi l'answer. Questa decisione sarà necessaria per eseguire correttamente il posizionamento del secondo interruttore del Modem. Se UNO è posizionato su ORIG l'altro dovrà spostare la levetta su Answ, oppure viceversa.

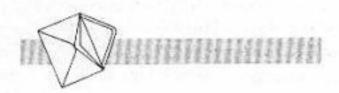
E' bene precisare che questa decisione non influenzerà la disposizione di chi dovrà trasmettere o su chi dovrà ricevere, ma sarà solamente una decisione inerente ad una necessità elettronica.

Solo a questo punto si potrà spostare la levetta TEL-MOD in posizione MOD. Ciò fatto si udirà nella cornetta una coppia di fischi continui e se la linea telefonica non è eccessivamente distribuita, la lampadina spia (LED) si accenderà. Da questo momento in poi ogni scambio di messaggi e dati sarà possibile. Il basso costo del Modem in ogni caso ci permette di creare una valida rete di comunicazioni.

Non ultima per importanza è la possibilità di comunicazione con banche dati già esistenti il cui elenco con relativi numeri telefonici e password, saranno pubblicati in un prossimo articolo che apparirà sulla rivista Commodore.

64 agricolo

☐ **È possibile applicare il C-64 in campo zootecnico e, in particolare, in un'azienda agricola? (*F. Carta - Arborea*)**

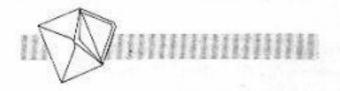
● Non siamo a conoscenza di software house che abbiano sviluppato programmi specifici. Pubblicheremo, comunque, il tuo annuncio tra le richieste di consulenza.

Espansioni

☐ **Esiste un'espansione di memoria per il Commodore 64? (*E. Venchiarutti - Cesano Boscone*)**

● Un microprocessore può usufruire di una quantità di memoria RAM oppure ROM determinata dal numero massimo di locazioni indirizzabili. I microprocessori ad otto bit, tra cui il 6510 di cui è dotato il 64, hanno, in genere, una capacità di indirizzamento a 16 bit e, di conseguenza, il numero di locazioni utilizzabili è dato dal risultato di 2 elevato alla 16ma potenza: 65536 byte.
Tale valore viene generalmente abbreviato con 64K dato che per 1K si intende un blocco di 1024 byte. Il Commodore 64 dispone già di 64K di memoria ed anzi ne possiede una quantità maggiore cui è possibile accedere mediante alcuni "trucchi" di programmazione in linguaggio macchina. Con trucchi analoghi (e ricorrendo ad una manciata di circuiti integrati) sarebbe possibile indirizzare un numero praticamente infinito di blocchi di memoria ma tale struttura creerebbe altri problemi di non facile soluzione.
Molti, erroneamente, pensano che la "potenza" di un computer dipenda in gran parte dalla capacità di memoria indirizzabile dimenticando che un linguaggio ben strutturato e un'architettura efficiente sono di gran lunga preferibili ad una gran massa di dati "in linea". In conclusione, sul mercato non vi sono espansioni di memoria per il C-64 perché, così come è posto in vendita, è già fin troppo pieno di ROM e di RAM. Discorso diverso vale per il Vic 20 e per il C-16 che, dotati di una quantità di memoria inferiore al massimo consentito dal microprocessore, presentano dei "buchi" colmabili, appunto, con espansioni di memoria.

Dischi laser

☐ **Sento molto parlare di videodischi o dischi laser. Esiste qualche versione collegabile al Commodore 64? (*D. Gallé - Serra san Bruno*).**

● Il video disco è frutto di una tecnologia che consente di immagazzinare una mole considerevole di dati in uno spazio ristretto. Molti sono stati gli sforzi dei costruttori per ottenere prodotti di basso costo in modo da realizzare, tra l'altro, enciclopedie su disco da usare insieme con TV domestico. Purtroppo si è ancora in fase sperimentale e alcune ditte non hanno ottenuto i risultati di vendita che si aspettavano. Ora come ora, non solo non è facile reperire gli apparecchi ma, a maggior ragione, non ve ne sono per il tuo personal computer.

Stoplist

☐ **Come si adopera, in pratica, il programma Stoplist pubblicato sul N. 14? (*A. Baldissera - Robegano*).**

● Se vuoi utilizzare la brevissima routine, devi caricare il programma pubblicato (ogni volta che accendi il Commodore 64) e battere RUN. Dopo aver fatto ciò, cancella pure il listato (NEW) e carica i programmi Basic che desideri: tutte le volte che chiederai il listato (LIST), questo verrà visualizzato come di consueto, a meno che tu non prema il tasto SHIFT.

In questo caso, infatti, la visualizzazione del listato si bloccherà e riprenderà nel momento in cui rilascerai il tasto stesso.

Vic 20 cecchino

☐ **In alcuni giochi ove occorre "sparare" mediante la pressione del tasto Fire del joystick, il mio Vic 20 spara continuamente anche se il tasto non viene premuto. Come mai? (*P. Angiuli -Bari*).**

● Devi provare a caricare lo stesso programma sul Vic di un tuo amico. Se il difetto persiste vuol dire che il programma (o il nastro) è difettoso. In caso contrario vuol dire che il tuo sistema presenta inconvenienti. Se il difetto scompare staccando il joystick vuol dire che l'accessorio è in corto circuito. In caso contrario è il connettore del Vic ad essere in corto. Semplice, no?

Incompatibilità IEEE-488

☐ **Collegando al Commodore 64 l'interfaccia IEEE-488 in modo da utilizzare la stampante 1361, molti programmi (Simon's Basic, Magic Desk ecc.) non girano più. C'è un rimedio? (*F. Pucci - Angri*).**

● No. L'interfaccia utilizza infatti, al momento stesso dell'inserimento, alcune locazioni di memoria vitali per quei programmi che, privi delle preziose informazioni e puntatori, non possono funzionare correttamente in tutto o in parte.

Scherzi da "INPUT".

☐ **Vorrei dei chiarimenti sul comando INPUT# poiché spesso riscontro delle difficoltà e dei messaggi di errore sui file, in programmi apparentemente corretti e dopo che la memorizzazione dei medesimi dati con PRINT# è avvenuta senza errori. Come separatore uso CHR$ (13) dopo ogni variabile. (*F. Iotti - Reggio Emilia*).**

● Cominciamo dal dire che mettere il CHR$ (13) dopo ogni variabile (o dato) è assolutamente inutile poiché ci pensa il computer ad inserire il "13" se non figura il carattere di punto e virgola dopo il tasto inviato. In altre parole i seguenti due comandi:

```
PRINT# 1, A; CHR (13);
PRINT# 1, A
```

producono lo stesso effetto, vale a dire la-

indirizzi) ed effettuare le correzioni del caso. Non esiste, infatti, una regola universale per rendere compatibili i software dei due sistemi.

Tieni comunque presente che se si tratta di giochi in alta risoluzione o di programmi che utilizzano sprite ed effetti sonori, la compatibilità è pressocché irrealizzabile. In ogni caso, per "trasportare" un programma registrato con un Commodore 64 su un computer 8000, 8096 oppure 4000, 3000, occorre seguire la procedura descritta sul N. 7 di C.C.C. nell'articolo "Ed ora passo al PET".

Strani scherzi del 1541

☐ **È possibile che il drive 1541 modifichi "da solo" i dati registrati in precedenza su disco? (*G. Robertazzi - S. Piero a Sieve*).**

● Assolutamente no, diamine! Se un computer non può prendere iniziative, figuriamoci una sua periferica, se pur "intelligente". Scherzi a parte: se il difetto lamentato riguarda dati (e non programmi), sei sicuro che, lavorando con i file, non ti succeda di modificarli, magari senza che te ne renda conto?
Se invece noti alterazioni nei programmi Basic è probabile che, nel far "ordine" sul dischetto, ti capiti di cancellare le versioni più recenti di un listato conservando, invece, le più vecchie. Se ritieni che i due casi precedenti siano da scartare, purtroppo non c'è che una risposta; il 1541 è difettoso.

Trasferimenti

☐ **Ho acquistato il drive 1541 e mi piacerebbe riportare su disco i numerosi giochi (protetti) che ho acquistato in precedenza su cassetta. È possibile? (*M. Pozzesi - Modena*)**

● Certamente, sempre che tu sia in grado di rimuovere la protezione...

Una o due riviste?

☐ **Perché sdoppiare in due riviste (Commodore & C.C.C.) articoli ed argomenti che riguardano le stesse macchine? Non potevate realizzare un'unica rivista più corposa (e costosa) ma anche più funzionale?**

● Il numero di utenti Commodore è talmente vasto ed assortito che non solo abbiamo ritenuto opportuno realizzare due pubblicazioni, ma abbiamo anche affiancato loro una rivista su cassetta (Commodore Club) per accontentare gli utenti "pigri".
Su Commodore Computer Club troverai argomenti semplici, affrontati in modo elementare e (me lo auguro) comprensibile anche per chi è alle prime armi. Commodore, viceversa, è destinato ad un pubblico più "evoluto" mentre l'acquirente di Commodore Club troverà i giochi, le utility ed i "professionali" troppo lunghi da affrontare senza errori di digitazione da rivista, ma troppo interessanti per non utilizzarli lasciandoli su carta...

Imprecisione o difetto?

☐ **Provando ad eseguire la sottrazione 6.1-6 si ottiene un risultato strabiliante. Come pure, battendo 9↑2, il risultato non è 81. Come mai? (*G. Matteo - Fontegreca. G. Ornaghi - Chieti*)**

● L'inconveniente non è dovuto ad un difetto della macchina ma alla routine di calcolo che rappresenta un "giusto" compromesso tra occupazione di memoria ROM, velocità di esecuzione di calcoli e semplicità di architettura del sistema. Non dimentichiamo che il Commodore 64 (come pure il Vic 20 ed il C-16) è un personal computer della fascia bassa che, per quello che costa, non può offrire la precisione disponibile su calcolatori ben più costosi.
Il secondo inconveniente lamentato (elevazione a potenza) può facilmente esser evitato ricorrendo al calcolo 9*9 invece che a 9↑2. Per ciò che riguarda il primo, invece è indispensabile crearsi una routine di approssimazione che, tenendo conto di un "epsilon" prefissato, consenta di aggirare l'imprecisione lamentata. Tale routine è decisamente più complessa di quanto possa sembrare a prima vista dato che deve funzionare in tutti i casi di difficile predeterminazione. C'è qualche lettore che voglia imbarcarsi nell'impresa?

Digitazione in fasi successive

☐ **Come devo fare per continuare a digitare un programma trascritto solo per metà? (*G. Moretti - Brescia*)**

● Se il programma che devi trascrivere da una rivista è composto da un elevato numero di linee, ad esempio 300, puoi iniziare a digitare le prime 50, registrarle su disco o nastro e verificare il caricamento (VERIFY). Il giorno dopo, o quando ti pare, ricaricherai tale "spezzone" col solito LOAD e, in seguito a tale operazione, continuerai a digitare le linee che mancano ripetendo il sistema descritto fino a completamento dell'operazione.
Se, invece, siete un gruppo di amici e desiderate ripartirvi il lavoro, ognuno di voi digiterà, registrerà e verificherà un gruppo di linee che in seguito potranno esser "fuse" tra loro con il programma APPEND pubblicato sul N. 15 di C.C.C.
Per evitare equivoci, tieni presente che un programma digitato per metà non può funzionare "a metà" come riteneva — ingenuamente — un lettore alle prime armi che si lamentava per il mancato funzionamento di un listato digitato, appunto, solo in parte.

Caratteri all'accensione

☐ **Perché al momento dell'accensione del mio Commodore 64, appaiono alcuni caratteri distribuiti variamente sullo schermo? (*G. Ornaghi - Chieti*)**

● Non appena si fornisce tensione al computer, viene attivata una routine che provvede ad eseguire numerosissimi compiti tra cui la trascrizione degli appropriati puntatori, il colore del bordo e del fondo, l'azzeramento degli sprite, eccetera. Tra queste, figura anche la routine di cancellazione schermo che, a quanto pare, non funziona correttamente sul tuo apparecchio. Ciò è dovuto al fatto, probabilmente, che le RAM dello schermo non funzionano immediatamente non appena viene data tensione e, di conseguenza, quando la routine

INVIARE TUTTA LA PAGINA ANCHE SE SI UTILIZZA UNA SOLA SCHEDA

Nome .... **Cognome** ....

Via .... **N°** .... **CAP.** .... **Città** ....

Telefono .... **Orario** ....

Registrate il mio abbonamento annuale a Commodore Computer Club.

☐ Ho versato oggi stesso il canone di L. 28.000 a mezzo c/c postale n° 31532203 intestato a:
Commodore Computer Club - V.le Famagosta, 75 - 20142 Milano

☐ Ho inviato oggi stesso assegno bancario n. ....
per l'importo di L. 28.000 intestato a Commodore Computer Club.

Si prega di scrivere il proprio nome e l'indirizzo completo in modo chiaro e leggibile. Inviare la fotocopia del bollettino di c/c postale.

Considerando che i numeri 1, 2 e 7 sono esauriti, vogliate inviarmi i numeri arretrati .... al prezzo di L. 5.000 cadauno per richieste fino a 4 numeri, o di L. 4.000 cadauno per richieste oltre i 4 numeri arretrati, e perciò per un totale di L............... Sono a conoscenza che i fascioli suddetti non saranno inviati in contrassegno e, pertanto, ho provveduto oggi stesso a versare il canone di L................ a mezzo c/c postale n. 31532203 intestato a:
Commodore Computer Club - V.le Famagosta, 75 - 20142 Milano

STATISTICA

Non posseggo un computer ☐

Posseggo un C64 .... si ☐ ... no ☐
Posseggo un VIC 20 .... si ☐ ... no ☐
Posseggo un Commodore Plus 14 .... si ☐ ... no ☐
Posseggo un Commodore Plus 16 .... si ☐ ... no ☐
Posseggo un registratore dedicato .... si ☐ ... no ☐
Posseggo un drive 1541 .... si ☐ ... no ☐
Posseggo una stampante .... si ☐ ... no ☐
Posseggo un monitor .... si ☐ ... no ☐

COLLABORAZIONE

A titolo di prova vi invio un articolo e la cassetta.........................disco....................................
col programma che intendo proporre per la pubblicazione di cui garantisco l'originalità.

DOMANDA/RISPOSTA ....

....

....

....

....

....

RICHIESTA ARGOMENTI

Mi farebbe piacere che Commodore Computer Club parlasse più spesso dei seguenti argomenti:

1/ ....

2/ ....

3/ ....

4/ ....

GIUDIZIO SUI PROGRAMMI DI QUESTO NUMERO

Ho assegnato un voto da 0 a 10 ai programmi che indico di seguito:

A/ .... Voto ....

B/ .... Voto ....

C/ .... Voto ....

D/ .... Voto ....

PICCOLI ANNUNCI ....

....

....

....

....

....

CERCO/OFFRO CONSULENZA ....

....

....

....

....

....

INVIARE IN BUSTA CHIUSA E AFFRANCANDO SECONDO LE TARIFFE VIGENTI A:

COMMODORE COMPUTER CLUB

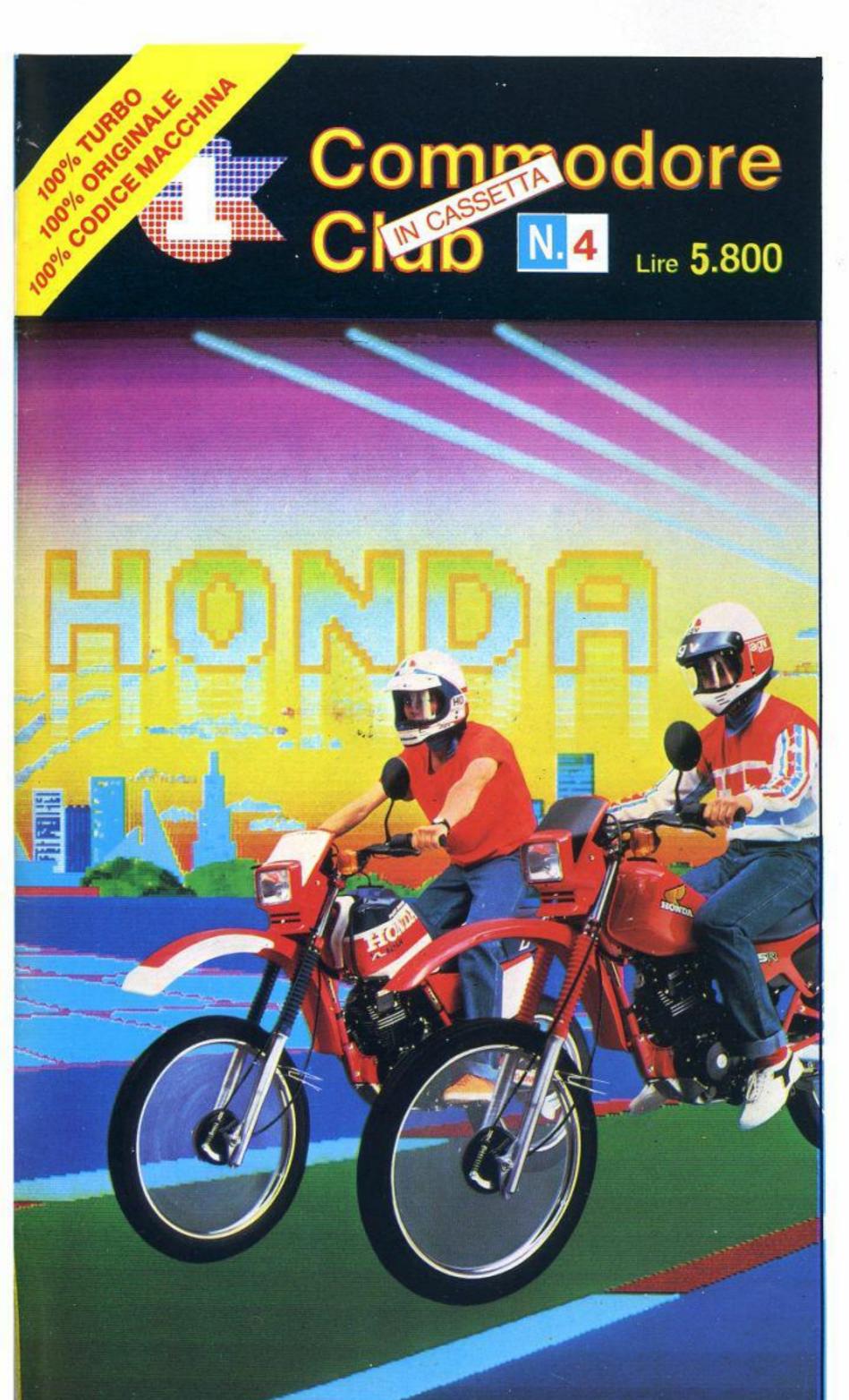
V.le Famagosta, 75
20142 Milano

INVIARE TUTTA LA PAGINA ANCHE SE SI UTILIZZA UNA SOLA SCHEDA

Nome .... **Cognome** ....

Via .... **N°** .... **CAP.** .... **Città** ....

Telefono .... **Orario** ....

Una cassetta eccezionale! da chiedere alla tua edicola

- **Honda** (C64)
- **Elettricista** (C64)
- **Geppetto** (C64)
- **Batteria** (C64)
- **Routine grafiche** (C64)
- **Salterello** (Vic 20)
- **Attacco** (Vic 20)
- **Racchette** (Vic 20)
- **Istogrammi** (Vic 20)
- **Caleidoscopio** (C16/+4)
- **Dama** (C16/+4)
- **Chiudi la porta** (C16/+4)